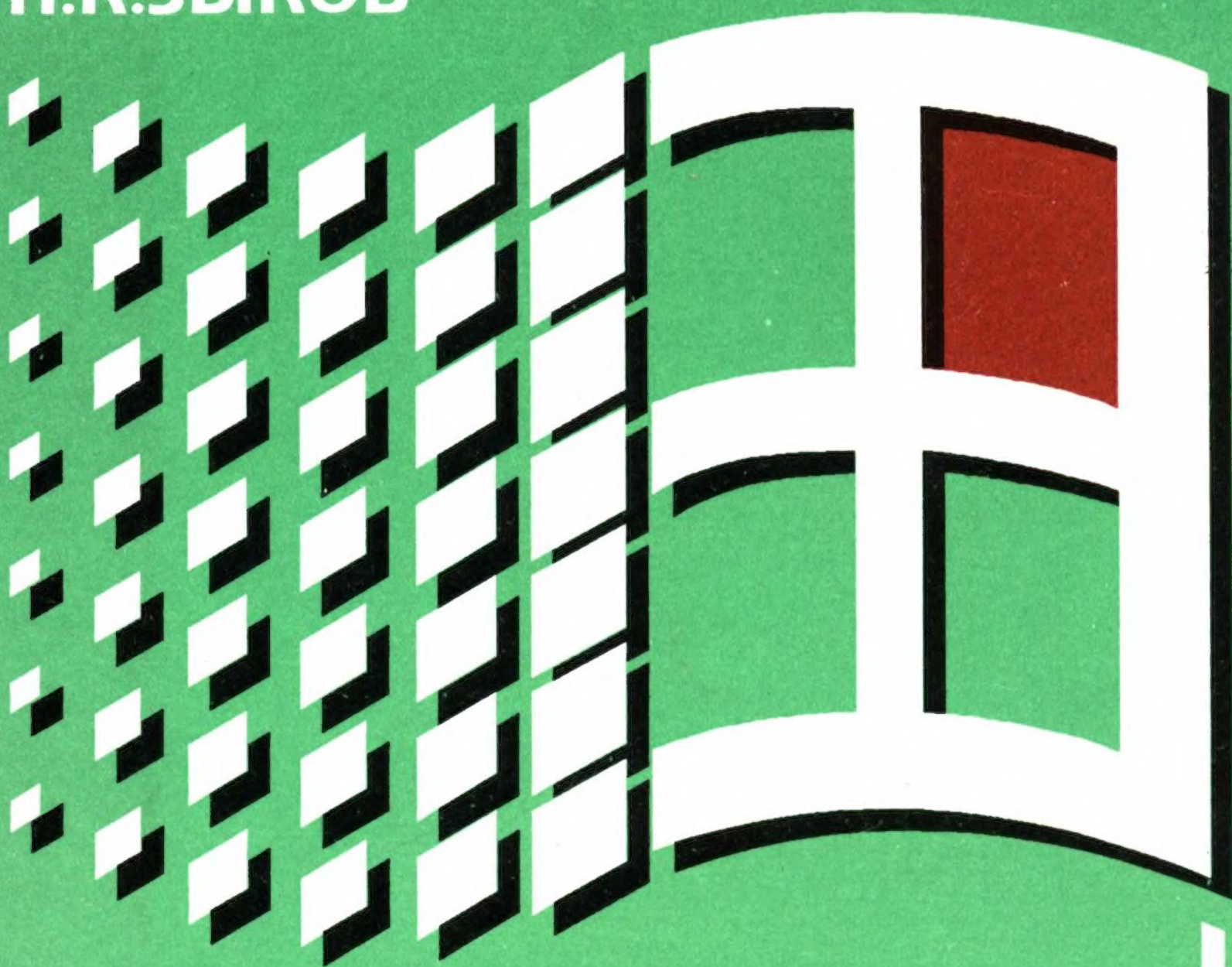


Н.К.ЗЫКОВ



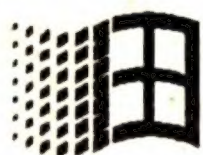
**Недокументированные
возможности**

WINDOWS



Н.К.ЗЫКОВ

Недокументированные



ВОЗМОЖНОСТИ

WINDOWS

справочник
для программиста-практика

МОСКВА
РАДИО И СВЯЗЬ
1994



ББК 32.97
3-96
УДК 681.3

Зыков Н. К.

3-96 Недокументированные возможности Windows: Справочник для программиста-практика. — М.: Радио и связь, 1994. — 176 с.

ISBN 5-256-01212-6.

Справочник содержит сведения о тех функциях, структурах и сообщениях операционной оболочки Windows версий 3.0 3.1, которые не были описаны в официальной документации, но могут быть эффективно использованы в прикладных программах.

Для программистов.

3 $\frac{1404000000-048}{046(01)-94}$ Без объявл.

ББК 32.97

Справочное издание

Зыков Никита Кириллович

Недокументированные возможности Windows

Справочник

Редактор И. Н. Сулова
Обложка В. Г. Ситникова
Корректор Т. Г. Тертышная

ИБ N 2623

ЛР № 010164 от 04.01.92

Подписано в печать с оригинал-макета 10.07.94
Гарнитура Таймс Печать офсетная Усл. печ. л. 10,78
Тираж 30000 экз. Изд. № 23892 Зак. № 3370 С-048

Формат 60x88/16
Усл. кр.-отт. 11,03

Бумага газетная
Уч.-изд. л. 6,41

Издательство "Радио и связь", 101000, Москва, Почтамт, а/я 693

Тираж отпечатан в Ордена Трудового Красного Знамени Чеховском полиграфическом комбинате Комитета Российской Федерации по печати. 142300, г.Чехов, Московской обл.

ISBN 5-256-01212-6

© Зыков Н. К., 1994
© Оформление и оригинал-макет.
Зыков Н. К., 1994

Предисловие

За последние годы Windows практически стали стандартом операционной оболочки для IBM-совместимых персональных компьютеров. В России распространение системы идет не так быстро, и тем не менее даже здесь Windows становятся все более популярными. Серьезные программисты заинтересовались этой средой и в последнее время в нашей стране для нее появилось много хороших программ.

Windows должны были стать альтернативой DOS (точнее, надстройкой, делающей DOS настоящей операционной системой), позволяющей писать программы, взаимодействующие с «внешним миром» средствами, предоставляемыми самой системой, без необходимости использования ее внутренних (недокументированных) возможностей и без прямого обращения к аппаратуре. По замыслу Windows должны позволять писать надежные и переносимые программы, не заботясь ни о подробностях выполнения запросов программы к системе, ни даже о типе конкретной машины. Но нередко возникает, например, необходимость оптимизировать низкоуровневое взаимодействие с внешними устройствами, и тогда, будучи практически лишен информации о системных объектах, программист не может организовать по-настоящему эффективное функционирование программы.

Существует и другая проблема: в *любой* достаточно сложной системе должны оставаться какие-то недокументированные возможности (структуры данных или функции), которые предназначены для реализации документированного взаимодействия с пользователем и программистом. Теоретически эти средства должны использоваться *только* ядром операционной системы и быть не пригодны для применения где-либо еще. На практике же многие из не описанных в документации функций и структур интенсивно используются не только теми модулями ядра системы, которым они не принадлежат, но и *прикладными* программами ряда крупных фирм (и, в частности, самой Microsoft).

С появлением версии Windows 3.1 положение немного улучшилось. Несколько ранее недокументированных функций (примерно 10...20 % от общего их числа) были описаны в

документации в рамках стратегии Open Tools. Это в основном касалось средств для создания отладчиков и фрагментов инициализации программ, причем многие очень полезные функции так и остались недокументированными. Также появилась совместимая с Windows 3.0 библиотека ToolHelp, дающая информацию о состоянии некоторых системных структур, но не предоставляющая практически никаких возможностей их изменения. Средства ToolHelp достаточно удобны, однако они охватывают лишь небольшую часть внутренних структур системы и к тому же, предоставляя довольно много информации, не дают никакого представления о реальном виде и функционировании этих структур.

Настоящий позволит вам восполнить большинство этих пробелов. В нем содержатся сведения об основных недокументированных структурах, переменных, сообщениях и функциях, которые могут быть эффективно использованы в прикладных программах. В справочник не включены действительно *внутренние* функции (пригодные только для использования ядром системы), функции-«затычки», оставшиеся от старых версий или нереализованные, а также функции, которые имеют прямые документированные аналоги. Материал, представленный в справочнике, не содержит информации о версиях Windows 1.x и 2.x. В нем также нет сведений о возможностях малопригодного для использования реального режима, остававшегося в версии 3.0.

Основным источником информации для данного справочника явилась книга Эндрю Шалмана (у нас его называют «по-немецки» Шульманом) «Undocumented Windows». В то же время, данный справочник не является переводом этого произведения. Книга Шалмана — это прежде всего исследовательская работа, объясняющая программисту тонкости функционирования системы и все аспекты (от юридических до философских) использования ее недокументированных возможностей (поэтому французский вариант названия этой книги — «Les coulisses de Windows» — «За кулисами Windows» гораздо точнее английского). Предлагаемый вам справочник является переработкой (с учетом опыта автора в программировании и исследовании системы) содержащейся в этой книге (равно как и

во многих других) информации, предназначенной в основном для использования в практическом программировании.

Справочник предназначен прежде всего для достаточно опытных Windows-программистов, которые хотят использовать в своих программах богатые возможности недокументированной части SDK этой системы. Все описания структур и функций в настоящей книге выполнены в стандарте языка C++, но программисты, пишущие на языках C и Pascal, также смогут использовать предоставленные здесь материалы.

Много полезного из справочника смогут извлечь хэкеры (программисты-исследователи), которым недоступна книга «Undocumented Windows» и специализированные сетевые конференции. Хотелось бы надеяться, что те представители вышеупомянутого вида искусства, к которым попадет данный справочник, не будут использовать его материалы для разного рода компьютерных хулиганств типа написания вирусов.

Справочник состоит из четырех частей, где в алфавитном порядке описаны недокументированные структуры, сообщения, экспортируемые переменные и функции Windows. В приложении предлагается достаточно условная классификация всех этих объектов по «темам».

Сведения о внутренних структурах приведены в основном для того, чтобы дать представление о реальном функционировании системы (существует довольно мало программ, из которых необходимо обращаться к ним напрямую). Дескрипторы сегментов локальных хипов¹ основных модулей Windows, необходимые для доступа к их локальным структурам, лучше всего получать средствами ToolHelp, но можно также использовать функции UserSeeUserDo() и GdiSeeGdiDo() (в версии 3.1), XCSToDS() (в версии 3.0) либо просто дескриптор экземпляра, возвращаемый функцией LoadLibrary(). Следует отметить, что таблица глобальных атомов (а в Windows 3.1 — также и меню) располагается в отдельном локальном сегменте данных USER.

Возможность реального использования недокументированных средств Windows существует благодаря тому, что модули ядра системы являются на самом деле динамическими библиотеками, содержащими экспортируемые функции и

¹ В русскоязычной литературе также встречаются названия «куча» и «локальная область памяти».

переменные. SDK, DDK и Windows-библиотеки языков программирования содержат только ссылки на соответствующие объекты, уже содержащиеся в системе. При этом так называемые недокументированные функции *не описаны только в файлах заголовков*. Все библиотеки импорта *содержат* соответствующие ссылки. Таким образом, для использования любой недокументированной функции или переменной достаточно поместить прототип функции в текст программы, а переменную объявить как *extern*.

Несмотря на то что использование недокументированных функций Windows гораздо менее опасно, чем это пытается представить фирма Microsoft, некоторая осторожность при их использовании все же не помешает. Для структур рекомендуется (если это возможно) проверять сигнатуры, а для функций — соответствие экспортируемого имени порядковому номеру в модуле. Вообще вместо прямого объявления прототипа соответствующей функции в тексте программы, лучше объявить указатель и получить адрес вызова функции следующей процедурой:

```
FARPROC RetrieveFunc(LPSTR szModule,WORD nOrdinal,
                    LPSTR szName=NULL){
    HANDLE hModule;
    if(! (hModule=GetModuleHandle(szModule)))
        if((hModule=LoadLibrary(szModule))
            <HINSTANCE_ERROR)
        return NULL;
    if(nOrdinal){
        FARPROC retval=
            GetProcAddress(hModule,
                LPSTR(MAKEULONG(nOrdinal,0)));
        if(retval &&
            (!szName || GetProcAddress(hModule,szName)
                ==retval)
            ) return retval;
        else return 0;
    }
    else{
        if(szName) return GetProcAddress(hModule,szName);
        else return 0;
    }
}
```

В качестве параметра `szModule` должно быть задано имя модуля, который экспортирует соответствующую функцию (в справочнике имя указывается перед точкой в разделе *Адрес*). Параметр `nOrdinal` задает номер экспортируемого входа (*Адрес*, после точки), причем нулевое значение позволяет подключать функцию только по имени. Необязательный параметр `szName` как раз и дает возможность проверять соответствие входа имени функции или подключать функцию по имени.

Приведенная выше функция позволяет также динамически подключать экспортируемые переменные, причем для двухбайтных переменных значение содержится в младшем слове возврата.

По возможности автор пытался использовать во всем справочнике унифицированную терминологию и соблюсти стандартные для Windows соглашения об описаниях. Чтобы строго следовать стандарту языка C++, пришлось в описаниях структур начало таблиц переменного размера обозначать как массив из одного элемента, что отражено в комментариях.

Поля структур и аргументы функций, значения которых не известны или сомнительны, обозначены как `reserved*` или `reserv*`. Значения соответствующих полей не должны изменяться, а аргументы функций (если обратное не оговорено) должны устанавливаться в 0.

Для описания различий в аргументах функций и структурах для различных версий и режимов Windows были использованы следующие директивы препроцессора:

```
#define MODE286      //Для стандартного режима
#define MODE386      //Для расширенного (386) режима

/*Две предыдущие директивы не могут вводиться
одновременно.*/

#define WIN30         //Windows 3.0
#define WIN31         //Windows 3.1
#define WIN31B        //Windows 3.1 beta

/*WIN30 и WIN31 совместно использоваться не могут;
WIN31B может использоваться только вместе с WIN31 */
```

Эти директивы имеют смысл только для объектов, определенных для версий Windows как 3.0, так и 3.1. Если вы хотите писать программы, предназначенные для использования с различными версиями Windows, то соответствующие

«динамические» варианты структур можно создать с помощью того же препроцессора.

В любом случае рекомендуется, если есть такая возможность, вместо недокументированных использовать соответствующие документированные средства, в частности библиотеку ToolHelp. Однако в определенных ситуациях вам не удастся этого сделать, из-за чего, собственно, и возникла необходимость создания справочника.

Основные области использования сведений справочника это:

создание асинхронных (независимых от очереди сообщений) процедур и управление асинхронными процессами;

манипуляция представлением ресурсов (иконки, курсоры и т.д.) в памяти;

фоновое наблюдение за изменениями в файловой системе;

реализация серверов Drag&Drop;

применение различных дескрипторов (HINSTANCE, HWND, HDC и т. п.) для получения полезной информации;

программная реализация для Windows 3.0 многих средств SDK версии 3.1, недоступных даже через ToolHelp.

Часть I

Внутренние структуры Windows

ATOMTABLE

Описание:

```
struct ATOMTABLE{  
    int nEntries;  
    ATOMENTRY near *pAtoms[];  
};
```

Поля:

nEntries	Число позиций в таблице.
pAtoms	Адрес хэш-таблицы, содержащей ссылки на элементы цепочки структур ATOMENTRY в локальной памяти.

Комментарии:

Структура описывает хэш-таблицу атомов, создаваемую в локальной памяти вызовом функции InitAtomTable(). Элементы таблицы — это либо указатели на соответствующие структуры ATOMENTRY, либо значения NULL.

Если таблица существует, то ее адрес можно получить из структуры INSTANCE. Структура глобальной таблицы полностью аналогична этой, но ее начальный размер и соответственно поле nEntries всегда равны 37.

Версии: 3.0, 3.1.

ATOMENTRY

Описание:

```
struct ATOMENTRY{  
    WORD wNext;  
    WORD wRef;  
    BYTE byLen;  
    BYTE Buffer[1];  
};
```

Поля:

wNext	Следующая структура ATOMENTRY в списке или NULL для последнего элемента.
wRef	Число обращений к атому.
byLen	Длина строки в байтах.
Buffer	Начало буфера, содержащего строку длиной wLen.

Комментарии:

АТОМ — короткий указатель на эту локальную структуру. Ее описание можно найти в файле Winexp.h, входящем в комплект SDK.

Версии: 3.0, 3.1.

BITMAPOBJ

Описание:

```
struct BITMAPOBJ{  
    GDIOBJHDR Header;  
    HANDLE hmemBitmap;  
    BOOL bSelected;  
    HANDLE reserved;  
};
```

Поля:

Header	Заголовок объекта (см. GDIOBJHDR).
--------	------------------------------------

hmemBitmap	Дескриптор блока памяти соответствующей структуры BITMAP.
bSelected	TRUE, если карта выбрана в одном из контекстов, иначе FALSE.

Комментарии:

HBITMAP — дескриптор блока памяти этой структуры в локальном хипе GDI.

Версии: 3.0, 3.1.

BRUSHOBJ

Описание:

```
struct BRUSHOBJ{
    GDIOBJHDR Header;
    LOGBRUSH logBrush;
    COLORREF crHatchBk;
};
```

Поля:

Header	Заголовок объекта (см. GDIOBJHDR).
logBrush	Соответствующая структура LOGBRUSH.
crHatchBk	Цвет фона для «штрихующей» кисти.

Комментарии:

HBRUSH — дескриптор блока памяти этой структуры в локальном хипе GDI.

Версии: 3.0, 3.1.

BURGERMASTER

Описание:

```
struct BURGERMASTER{
    BOOL bCheck;
```

```

        BOOL bFreeze;
        WORD wEntries;
#ifdef MODE286
        WORD nwFirst;
        WORD nwLast;
#else
        DWORD nwFirst;
        DWORD nwLast;
#endif
        BYTE byComp;
        BYTE byDestrLev;
#ifdef MODE286
        WORD nwDestrBytes;
        HANDLE hTable;
        HANDLE hFree;
        WORD wDelta;
        WORD pExpand;
        WORD pStats;
#else
        DWORD nwDestrBytes;
        WORD reserved[5];
#endif
        WORD wLRUAcess;
#ifdef MODE286
        WORD wLRUChain;
#else
        DWORD dwLRUChain;
#endif
        WORD wLRUCount;
#ifdef MODE286
        WORD nwDestrPara;
        WORD nwDestrChain;
#else
        DWORD nwDestrPara;
        DWORD nwDestrChain;
#endif
        WORD wFree;
};

```

Поля:

bCheck	Если TRUE, то структура не разрушена.
bFreeze	Если TRUE, то структура не может быть изменена или сжата.
wEntries	Число элементов в списке объектов глобального хипа (GLOBAL_ARENA).

nwFirst	Селектор (для расширенного режима — смещение внутри сегмента BURGERMASTER) первого объекта в списке с обратной связью.
nwLast	Селектор (для расширенного режима — смещение внутри сегмента BURGERMASTER) последнего объекта в списке.
byComp	Уровень сжатия структуры.
byDestrLev	Текущий уровень разрушения структуры.
nwDestrBytes	Текущее число недействительных байт таблицы.
hTable	Дескриптор заголовка списка таблиц дескрипторов.
hFree	Дескриптор заголовка списка таблиц свободных дескрипторов.
wDelta	Число дескрипторов, выделяемых по каждому запросу.
pExpand	Адрес для процедур обработки дескрипторов.
pStats	Адрес статистических таблиц.
wLRUAccess	Доступ к списку LRU на уровне прерываний.
wLRUChain	Адрес первого дескриптора в списке LRU.
dwLRUChain	Смещение внутри BURGERMASTER первого дескриптора из списка LRU ¹ .
wLRUCount	Число элементов в списке LRU.
nwDestrPara	Число параграфов для разрушимого кода.
nwDestrReserve	Ограничитель разрушимого кода.
wFree	Число свободных блоков.

Комментарии:

Структура определяет состояние глобальной памяти (хипа).

¹ *Last recently used* — дескрипторы тех структур GLOBAL_ARENA, к которым недавно обращалась система, нечто типа хэш-таблицы.

Основные (для программиста-практика) поля этой структуры — `nwFirst` и `nwLast`, адресующие начало и конец двунаправленного списка информационных блоков объектов глобального хипа (`GLOBAL_ARENA`), соответствующих MCB DOS.

Селектор этой структуры возвращается функцией `GlobalMasterHandle()`.

По возможности рекомендуется вместо этой структуры использовать соответствующие средства `ToolHelp`.

Версии: 3.0, 3.1.

CLASS

Описание:

```
struct CLASS{
    HANDLE hNext;
    WORD wSign;
    ATOM atomClass;
    HANDLE hDCE;
    WORD wNumWin;
    WNDCLASS wndClass;
    WORD wData[1];
};
```

Поля:

<code>hNext</code>	Адрес следующей структуры <code>CLASS</code> в сегменте данных <code>USER</code> .
<code>wSign</code>	Сигнатура 'NK'.
<code>atomCls</code>	Атом в сегменте данных <code>USER</code> , содержащий имя класса.
<code>hDCE</code>	Если для окон класса установлено <code>CS_CLASSDC</code> , то это дескриптор соответствующей структуры <code>DCE</code> , иначе <code>NULL</code> .
<code>wNumWin</code>	Текущее число окон этого класса.
<code>wndClass</code>	Копия соответствующей структуры <code>WNDCLASS</code> с обнуленными полями <code>lpzMenuName</code> и <code>lpzClassName</code> .
<code>wData</code>	Дополнительные данные экземпляра окна.

Комментарии:

Структура адресуется полем hClass структуры WND и описывает класс окна.

Версии: 3.0, 3.1.

CURSOR

Описание:

```
struct CURSOR{
    CURSORICONINFO ciiHeader;
    BYTE BitmapBits[1];
};
```

Поля:

ciiHeader	Информационный заголовок.
BitmapBits	Монохромная битовая матрица, задающая изображение курсора.

Комментарии:

HCURSOR — глобальный дескриптор этой структуры.

Версии: 3.0, 3.1.

CURSORICONINFO

Описание:

```
struct CURSORICONINFO{
    POINT pntHotSpot;
    WORD wWidth;
    WORD wHeight;
    WORD wBmSize;
    BYTE byPlanes;
    BYTE byBitsPix;
};
```

Поля:

pntHotSpot	Точка-указатель (hotspot) для курсора.
wWidth	Ширина битовой карты в пикселах.
wHeight	Высота битовой карты в пикселах.
wBmSize	Размер битовой карты в байтах.
byPlanes	Число битовых плоскостей в карте. Для курсора всегда 1.
byBitsPix	Число бит на пиксел в плоскостях. Для курсора всегда 1.

Комментарии:

Эта структура — единый заголовок представления в памяти ресурсов иконки и курсора, отличающаяся от соответствующих заголовков в дисковых файлах.

Версии: 3.0, 3.1.

DC

Описание:

```
struct DC{
    GDIOBJHDR Header;
    BYTE byFlags;
    BYTE reserved;
    HANDLE hMetaFile;
    HRGN hrgnClip;
    HANDLE hPDevice;
    HPEN hPen;
    HBRUSH hBrush;
    HFONT hFont;
    HBITMAP hBitmap;
    HPALETTE hPal;
    HANDLE hDevice;
    HRGN hrgnRaoClip;
    HANDLE hGDIInfo;
    HANDLE hPPen;
    HANDLE hPBrush;
    HANDLE reserv;
    HANDLE hPFont;
    LPVOID lpDevice;
```

```
WORD pLDevice;
WORD pRaoClip;
WORD pPDeviceBlock;
WORD pPPen;
WORD pPBrush;
WORD reserv2;
LPVOID lpFont;
int reserved2[2];
WORD wROP2;
WORD wBkMode;
DWORD dwBkColor;
DWORD dwTextColor;
int nTBreakExtra;
int nBreakExtra;
WORD wBreakErr;
int nBreakRem;
int nBreakCount;
int nCharExtra;
COLORREF crLBkColor;
COLORREF crTextColor;
POINT ptLCursorPos;
POINT ptWndOrg;
POINT ptWndExt;
POINT ptVportOrg;
POINT ptVportExt;
POINT reserved3;
WORD wMapMode;
WORD reserved4;
WORD wRelAbs;
WORD wPolyFillMode;
WORD wStretchBltMode;
BYTE byPlanes;
BYTE byBitsPix;
WORD wPenWidth;
WORD reserv3;
WORD wTextAlign;
DWORD dwMapperFlags;
POINT ptBrushOrg;
POINT ptFontAspect;
HANDLE hFontWeights;
WORD wDCSaveLevel;
WORD reserved5;
HRGN hVisRgn;
POINT ptDCOrg;
FARPROC lpfnPrint;
ATOM atomDCLog;
ATOM atomDCPhys;
ATOM atomDCFile;
```

```

        DWORD reserved6;
#ifdef WIN30
        WORD reserved7;
        RECT reserved8;
#else
        RECT reserved7;
        RECT rectLVB;
        LPVOID reserved8[2];
#endif
        WORD wDCGlobalFlags;
#ifdef WIN31B
        HDC hDCNext;
#endif
};

```

Поля:

Header	Заголовок DC как GDI-объекта.
byFlags	Битовые флаги. Установленный бит 2 означает, что VisRgn изменился и требуются пересчет и (возможно) перерисовка DC.
hMetaFile	Если на DC проигрывается метафайл, то это его дескриптор.
hrgnClip	Прямоугольная область отсечения DC.
hPDevice	Дескриптор информационной структуры PDEVICE, документированной в DDK.
hPen	«Перо», выбранное в контексте.
hBrush	«Кисть», выбранная в контексте.
hFont	Шрифт, выбранный в контексте.
hBitmap	Битовая карта, выбранная в контексте, или NULL.
hPal	Текущая палитра контекста или NULL.
hDevice	Дескриптор структуры DEVICE в локальном хипе GDI, содержимое которой не известно.
hrgnRaoClip	Видимая часть текущей области отсечения.
hGDIInfo	Дескриптор соответствующей структуры GDIINFO, документированной в DDK.

hPPen	Дескриптор физического «пера» (для драйверов устройств).
hPBrush	Дескриптор физической «кисти».
hPFont	Дескриптор физического шрифта (возвращается функцией <i>GetPhysicalFontHandle()</i>).
lpPDevice	Адрес структуры, на которую ссылается <i>dc.hPDevice</i> .
pLDevice	Результат <i>LocalLock(dc.hPDevice)</i> .
pRaoClip	Результат <i>LocalLock(dc.hrgnRaoClip)</i> .
pPGDIInfo	Результат <i>LocalLock(dc.hGDIInfo)</i> .
pPPen	Результат <i>LocalLock(dc.hPPen)</i> .
pPBrush	Результат <i>LocalLock(dc.hPBrush)</i> .
lpPFont	Адрес структуры, на которую ссылается <i>dc.hPFont</i> .
wROP2	Текущий режим отрисовки, задаваемый <i>SetROP2()</i> .
wBkMode	Текущий режим заполнения фона.
dwBkColor	Внутреннее (для драйвера) представление текущего цвета фона.
dwTextColor	Внутреннее представление текущего цвета символов текста.
nTBreakExtra	Значение, устанавливаемое параметром <i>nBreakExtra</i> функции <i>SetTextJustification()</i> .
nBreakExtra	Число пикселей выравнивания для вывода через <i>TextOut()</i> . Вычисляется как округленное частное полей <i>nTBreakExtra</i> и <i>nBreakCount</i> .
wBreakErr	Код ошибки, возвращаемый, если параметр <i>nBreakCount</i> функции <i>SetTextJustification()</i> установлен в 0.
nBreakRem	Остаток от целочисленного деления <i>nTBreakExtra</i> на <i>nBreakCount</i> .
nBreakCount	Значение, устанавливаемое параметром <i>nBreakCount</i> функции <i>SetTextJustification()</i> .

nCharExtra	Значение, устанавливаемое параметром nCharExtra функции SetCharacterExtra().
crBkColor	Логический цвет фона.
crTextColor	Логический цвет символов.
ptLCursorPos	Координаты текущей позиции пера.
ptWndOrg	Начало координат окна.
ptWndExt	Ширина и высота окна.
ptVportOrg	Начало координат области вывода.
ptVportExt	Ширина и высота области вывода.
wMapMode	Текущий режим преобразования координат.
wRelAbs	Режим задания координат устройства (абсолютные/относительные).
wPolyFillMode	Режим заполнения многоугольников.
wStretchBltMode	Тип сжатия для функции StretchBlt().
byPlans	Число битовых плоскостей для последней выбранной в контексте битовой карты.
byBitsPix	Число бит плоскости на пиксел для последней выбранной в контексте битовой карты.
wPenWidth	Толщина текущего «пера».
wTextAlign	Флаги выравнивания текста.
dwMapperFlags	Параметры преобразования шрифтов.
ptBrushOrg	Координаты начальной точки для текущей кисти.
ptFontAspect	Растяжение шрифта по горизонтали и по вертикали.
hFontWeights	Жирность шрифта. Часто игнорируется.
wDCSaveLevel	Число вызовов SaveDC(), не скомпенсированных RestoreDC().
hVisRgn	Видимая область DC.
ptDCOrg	Начало координат DC.
lpfnPrint	Точка входа для драйвера принтера.
atomDCLog	Локальный атом GDI, содержащий логическое имя устройства, на котором выделен DC.

atomDCPhys	Локальный атом GDI, содержащий физическое имя соответствующего устройства.
atomDCFile	Атом, содержащий имя файла, куда перенаправлена печать, либо NULL.
rectLVB	Область DC, нуждающаяся в обновлении.
wDCGlobalFlags	Если бит 0 установлен, DC нуждается в обновлении.
hDCNext	Следующий DC в списке.

Комментарии:

DC — структура локального хипа GDI, на которую указывает HDC.

Версии: 3.0, 3.1.

DCE

Описание:

```
struct DCE{
    HANDLE hDCENext;
    HWND hWndCurr;
    HDC hDC;
    BYTE byFlags;
    BYTE byInUse;
    BYTE byDirty;
    BYTE reserved;
    POINT ptOrigin;
    HWND hwndTop;
    HRGN hVisRgn;
};
```

Поля:

hDCENext	Дескриптор следующей структуры DCE.
hWndCurr	Окно, использующее DC, или NULL.
hDC	Дескриптор DC, обслуживаемого структурой.

byFlags	Бит 1 установлен, когда DC обслуживает рабочую область окна, бит 2 — все окно.
byInUse	0, если DCE свободна.
byDirty	1, если требуется перерисовка части DC.
ptOrigin	Начало координат DC.
hwndTop	Окно верхнего уровня — предок hwndCurr.
hVisRgn	Видимая область окна.

Комментарии:

Структура служит для организации связной цепочки DC в локальном хипе USER.

Версии: 3.0, 3.1.

DCSAVE

Комментарии:

DCSAVE — подмножество структуры DC, используемое при сохранении/восстановлении состояния (SaveDC()/RestoreDC(), GetDCState()/SetDCState()). Структура состоит из полей DC от 'Header' до wDCLocks и поля ptDCOrg.

Версии: 3.0, 3.1.

DRAGINFO

Описание:

```
struct DRAGINFO{
    HWND hwndSource;
    HANDLE reserved;
    WORD wFlags;
    WORD pList;
    HANDLE hOfstruct;
    POINT ptCoord;
    DWORD reserved2;
};
```

Поля:

hwndSource	Окно, из которого начато перемещение (Drag).
wFlags	Тип перемещаемых объектов.
pList	Указатель в сегменте данных приложения-источника на список имен перемещаемых объектов, разделенных пробелами.
hOfstruct	Если перемещается файл, расширение которого имеет привязанную программу в файле Win.ini, то дескриптор соответствующей структуры OFSTRUCT, иначе NULL.
ptCoord	Позиция объекта на поверхности окна клиента.

Комментарии:

Структура используется сообщениями протокола Drag&Drop. wFlags может иметь следующие значения:

1 (DRAGOBJ_PROGRAM) — имя файла, расширение которого указано в разделе «Programs=» Win.ini;

2 (DRAGOBJ_DATA) — имя непривязанного файла данных;

3 (DRAGOBJ_DIRECTORY) — имя подкаталога;

4 (DRAGOBJ_MULTIPLE) — список файлов разных типов.

Если окно-сервер и окно-клиент принадлежат разным программам или экземплярам программы, то также устанавливается старший бит wFlags ($wFlags |= 0x8000$, что в данном случае эквивалентно $wFlags += 0x8000$).

Версии: 3.0, 3.1.

DROPINFO

Описание:

```
struct DROPINFO{  
    WORD wOfsFirst;  
    POINT ptDrop;  
    BOOL bClient;
```

```

        BYTE byBuffer[1];
};

```

Поля:

wOfsFirst	Смещение chBuffer от начала структуры.
ptDrop	Точка рабочей области, в которой был произведен «сброс» (Drop).
bClient	TRUE, если «сброс» был произведен в рабочей области, иначе FALSE.
byBuffer	ASCIIZ-буфер, содержащий разделенный пробелами список имен перемещаемых объектов.

Комментарии:

DROPINFO — та самая глобальная «внутренняя структура», дескриптор которой передается в wParam с сообщением WM_DROPFILES.

Версия: 3.1.

FONTOBJ

Описание:

```

struct FONTOBJ{
    GDIOBJHDR Header;
    LOGFONT logfont;
};

```

Поля:

Header	Заголовок шрифта как объекта GDI.
logfont	Соответствующая структура LOGFONT.

Комментарии:

HFONT — дескриптор этой структуры в локальном хипе GDI.

Версии: 3.0, 3.1.

GDIOBJHDR

Описание:

```
struct GDIOBJHDR{
    HANDLE hNext;
    WORD wType;
    DWORD dwNumber;
    WORD reserved;
#ifdef WIN31B
    WORD wSelCount;
    HANDLE hOwner;
#endif
};
```

Поля:

hNext	Следующий элемент в списке объектов.
wType	Тип объекта и некоторые флаги.
dwCount	Порядковый номер объекта в общем списке.
wSelCount	Счетчик селекции объекта в контексте устройства.
hOwner	Владелец объекта. Поле позволяет Windows проверять, освобождены ли выделенные объекты к моменту завершения задачи.

Комментарии:

Эта структура — общий тип заголовка для объектов GDI. Конкретный тип объекта можно определить, замаскировав биты 13 и 15 поля wType (*gdiobjhdr.wType &= 0xafff*) следующим образом:

*Значение для версий**Тип объекта*

3.0

3.1

1

'GO'

Карандаш (PEN)

2

'HO'

Кисть (BRUSH)

3

'IO'

Шрифт (FONT)

4

'JO'

Палитра (PALETTE)

5

'KO'

Битовая карта (BITMAP)

6

'LO'

Область (REGION)

7

'MO'

Контекст устройства (DC)

8

'NO'

Неактивный контекст

9	'OO'	MetaDC (?)
10	'PO'	Метафайл (METAFILE)
11	'QO'	Контекст метафайла в памяти

Бит 15 wType установлен, если объект выбран в одном из контекстов устройств, бит 13 устанавливается Windows 3.1, когда объект помечается задачей как «собственный» функцией MakeObjectPrivate().

Версии: 3.0, 3.1.

GLOBAL_ARENA

Описание:

```
struct GLOBAL_ARENA{
    #ifdef MODE386
        DWORD nwPrev;
        DWORD nwNext;
        DWORD dwBase;
        DWORD dwSize;
        HANDLE hThis;
        HANDLE hOwner;
    #endif
        BYTE byLocks;
    #ifdef MODE286
        HANDLE hOwner;
        WORD wSize;
    #else
        BYTE byPageLocks;
    #endif
        BYTE byFlags;
    #ifdef MODE286
        WORD nwPrev;
        WORD nwNext;
        HANDLE hThis;
        WORD nwLRUPrev;
        WORD nwLRUNext;
    #else
        BYTE byReqSels;
        DWORD nwLRUPrev;
        DWORD nwLRUNext;
    #endif
};
```

Поля:

nwPrev	Селектор предыдущей структуры в общем списке структур GLOBAL_ARENA. Первый и последний селекторы списка указывают сами на себя.
nwNext	Селектор следующей структуры в списке.
dwBase	База сегмента памяти, соответствующего данной структуре.
dwSize	Размер блока в байтах.
hThis	Дескриптор блока памяти, который описывает данная структура. Если блок фиксирован, то младший бит дескриптора равен единице.
hOwner	Дескриптор владельца блока (модуль, задача или PDB). Специальные значения этого поля: 0 для свободного блока, -1 для «независимого» блока (владелец — сам блок), -3 для блока, содержащего структуру BURGERMASTER, -4 для физически внешнего блока, -7 для временного блока.
byPageLocks	Число страничных блокировок функцией GlobalPageLock().
byFlags	Третий бит установлен для блока DGROUP (сегмент данных по умолчанию экземпляра задачи), четвертый — для сбрасываемого блока.
nwLRUPrev	Предыдущий селектор в цепочке LRU ¹ .
nwLRUNext	Следующий селектор в LRU-цепочке.
byReqSels	Число необходимых селекторов для huge (бóльших 64K) блоков.

¹ *Last recently used* — дескриптор тех структур GLOBAL_ARENA, к которым недавно обращалась система.

Комментарии:

GLOBAL_ARENA играет в Windows роль описателя блока глобальной памяти, что очень близко к роли MCB для DOS.

В стандартном (286) режиме блок памяти, описываемый структурой, физически располагается сразу за ней, т.е. его базовый адрес равен адресу его GLOBAL_ARENA плюс 16.

Адрес начала и конца цепочки этих структур можно извлечь из структуры BURGERMASTER.

По возможности рекомендуется вместо этой структуры использовать соответствующие средства ToolHelp.

Версии: 3.0, 3.1.

ICON

Описание:

```
struct ICON{
    CURSORICONINFO ciiHeader;
    BYTE BitmapBits[1];
};
```

Поля:

ciiHeader	Информационный заголовок.
BitmapBits	Битовая матрица, задающая изображение иконки.

Комментарии:

HICON — глобальный дескриптор этой структуры.

Версии: 3.0, 3.1.

INSTANCE

Описание:

```
struct INSTANCE{
    BOOL bDoesntExist;
```

```
    DWORD dwOldSSSP;  
    WORD pLocalHeap;  
    WORD pAtomTable;  
    WORD pStackTop;  
    WORD pStackMin;  
    WORD pStackBottom;  
};
```

Поля:

bDoesntExist	FALSE, если эта структура присутствует в сегменте данных.
dwOldSSSP	В этом поле сохраняется старый адрес стека при его переключении функцией SwitchStackTo().
pLocalHeap	Указатель на соответствующую структуру LOCAL_HEAP. При отсутствии у модуля локального хипа здесь может содержаться случайное значение, так что желательно всегда проверять сигнатуру структуры.
pAtomTable	Указатель на соответствующую структуру ATOMTABLE или 0 в случае ее отсутствия.
pStackBottom	Указатель на вершину стека.
pStackMin	Указатель на конец использованной части стека. Общий размер этой части равен (pStackBottom-pStackMin).
pStackTop	Указатель на вершину стека.

Комментарии:

Дескриптор экземпляра программы, передаваемый ей как аргумент hInstance функции WinMain(), содержит селектор сегмента данных экземпляра по умолчанию.

В первых шестнадцати байтах этот сегмент может содержать описанную структуру. Поле bDoesntExist позволяет проверять ее действительное наличие.

Версии: 3.0, 3.1.

LOCAL_ARENA

Описание:

```
struct LOCAL_ARENA{
    WORD pPrev;
    WORD pNext;
    union{
        struct{
            WORD wSize;
            WORD pPrev;
            WORD pNext;
        }free;
        struct{
            WORD wEntries;
            WORD pFirst;
            WORD pFirstFree;
        }first;
    }WORD wMovReserv;
};
```

Поля:

pPrev	Последний бит установлен, если блок не свободный, предпоследний — если блок перемещаем. Замаскировав два последних бита ($pPrev \&= (\sim 3)$), можно получить указатель на предыдущий блок. Для первого блока поле указывает само на себя.
pNext	Следующий блок в списке. Последний блок указывает сам на себя.
free.wSize	Размер свободного блока.
free.pPrev	Указатель на предыдущий элемент в списке свободных блоков.
free.pNext	Указатель на предыдущий свободный блок.
first.wEntries	Число элементов в списке блоков локальной памяти.
first.pFirst	Указатель на первый элемент в списке, т. е. на себя.

first.pFirstFree	Указатель на первый элемент в списке свободных блоков.
wMovReserv	Сразу за этим полем начинается область данных для перемещаемого (LMEM_MOVEABLE) блока памяти. Назначение его точно не известно.

Комментарии:

Структура описывает заголовок локального блока памяти. Первый блок в списке использует поля подструктуры first, все свободные блоки — поля free. В остальных (неперемещаемых) блоках область данных начинается сразу за полем pNext.

Блоки в цепочке расположены последовательно, так что размер блока можно вычислить, посчитав расстояние от начала области данных (конец структуры) до заголовка следующего блока, на который указывает поле pNext.

По возможности рекомендуется вместо этой структуры использовать соответствующие средства ToolHelp.

Версии: 3.0, 3.1.

LOCAL_HEAP

Описание:

```
struct LOCAL_HEAP{
    BOOL bValid;
    BOOL bFreeze;
    WORD wEntries;
#ifdef MODE286
    WORD nwFirst;
    WORD nwLast;
#else
    DWORD nwFirst;
    DWORD neLast;
#endif
    BYTE byComp;
    BYTE byDestrLevel;
#ifdef MODE286
    WORD nwDestrBytes;
#else
```

```

        DWORD nwDestrBytes;
    #endif
    struct{
        WORD wEntries;
        struct{
            WORD pArenaMov;
            BYTE byFlags;
            BYTE byLocks;
        }Blocks[1];
    }NEAR *pMovList;
    WORD pMovAvail;
    WORD wMovDelta;
    WORD pfnExpand;
    WORD pStatistics;
    FARPROC lpfnLocalNotify;
    WORD reserved;
    WORD wDSIncr;
    WORD wMinHeap;
    WORD wSign;
};

```

Поля:

bValid	TRUE, если структура не разрушена и может использоваться.
bFreeze	TRUE, если структура не должна быть сжата.
wEntries	Число элементов в списке объектов локального хипа.
nwFirst	Короткий указатель на первую структуру LOCAL_ARENA в списке с обратной связью.
nwLast	Короткий указатель на последний элемент в списке структур LOCAL_ARENA.
byComp	Уровень сжатия структуры.
byDestrLevel	Текущий уровень разрушения структуры.
nwDestrBytes	Число разрушенных байт.
pMovList	Указатель на таблицу, описывающую перемещаемые блоки локального хипа. Если таких блоков в хипе нет, поле содержит NULL.

Таблица перемещаемых блоков:

wEntries	Число элементов таблицы.
Blocks	Начало таблицы описаний блоков.

Элементы таблицы:

pArenaMov	Указатель на соответствующий блок памяти.
byFlags	Флаги типа памяти.
byLocks	Счетчик блокировок, 0 для неиспользуемого блока.
pMovAvail	Указатель на первый свободный элемент в таблице перемещаемых блоков.
wMovDelta	Число элементов в таблице перемещаемых блоков, выделяемых по запросу.
pfnExpand	Указатель на функцию распаковки структуры.
pStatistics	Указатель на статистическую таблицу.
lpfnLocalNotify	Дальний указатель на процедуру-наблюдатель хипа, устанавливаемую функцией LocalNotify(). Это реликт версий Windows 2.x.
wDSIncr	Минимальный инкремент размера сегмента данных.
wMinHeap	Минимальный размер хипа.
wSign	Сигнатура 'LH'.

Комментарии:

Структура определяет состояние локального хипа. Дескриптор этой структуры содержится в поле pLocalHeap структуры INSTANCE.

Перемещаемые локальные блоки в Windows обрабатываются специальным образом. Когда память выделяется функцией LocalAlloc() с установленным флагом LMEM_MOVEABLE, возвращается не обычный дескриптор, а короткий указатель на «свежевыделенный» элемент таблицы перемещаемых блоков (этот указатель может быть передан в качестве аргумента функции

LocalLock()). При этом в заголовке (LOCAL_ARENA) этого блока памяти маскируется бит 2 поля `rPrev`.

По возможности рекомендуется вместо этой структуры использовать соответствующие средства ToolHelp.

Версии: 3.0, 3.1.

MENU

Описание:

```
struct MENU{
    #ifdef WIN31
        HMENU hmenuNext;
    #endif
    WORD reserved;
    #ifdef WIN30
        WORD wCurrSel;
        WORD reserved2;
        WORD wTotalSize;
    #else
        WORD wSign;
        HANDLE hTaskQueue;
    #endif
    WORD wWidth;
    WORD wHeight;
    WORD wItems;
    HWND hwndOwner;
    #ifdef WIN30
        MENUITEM Items[1];
    #else
        HANDLE hItems;
        WORD reserved2;
    #endif
};
```

Поля:

<code>hMenuNext</code>	Дескриптор следующего меню в общем списке.
<code>wCurrSel</code>	Порядковый номер выбранного элемента меню или -1.

wTotalSize	Общий размер в байтах меню со всеми соответствующими структурами MENUITEM.
wSign	Сигнатура 'NK'.
hTaskQueue	Структура TASK_QUEUE задачи, которой принадлежит меню.
wWidth	Ширина (в пикселах) всей рамки меню.
wHeight	Высота (в пикселах) рамки меню.
wItems	Число элементов в меню (число минус один в версии Windows 3.0).
hwndOwner	Окно-владелец списка.
hItems	Дескриптор локального блока памяти, содержащего соответствующий массив структур MENUITEM.

Комментарии:

HMENU — локальный дескриптор (в версии 3.0 — короткий указатель) в локальной области данных USER для этой структуры. Вряд ли структура может иметь практическое применение в прикладной программе.

Версии: 3.0, 3.1.

MENUITEM

Описание:

```
struct MENUITEM{  
    WORD wFlags;  
    WORD wCommand;  
    RECT rectItem;  
    WORD wTab;  
    HANDLE hCheckBmp;  
    HANDLE hUnchBmp;  
    HANDLE hContents;  
    WORD wUlStart;  
    WORD wUlLen;  
    WORD wTextLen;  
};
```

Поля:

wFlags	Флаги MF_... для данного пункта.
wCommand	Идентификатор (команда) пункта меню или дескриптор рорир-подменю, открываемого данным пунктом.
rectItem	Прямоугольник (в экранных координатах), занимаемый данным пунктом.
wTab	Текущая позиция табуляции пункта.
hCheckBmp	Дескриптор битовой карты «галочки», используемой для отметки пункта меню.
hUnchBmp	Дескриптор битовой карты для отметки пункта с атрибутом MF_UNCHECKED. Обычно NULL.
hContents	Либо атом, содержащий текст пункта меню, либо дескриптор соответствующей битовой карты.
wUlStart	Абсцисса (в пикселах) начала подчеркивания для буквы, соответствующей «горячей клавише».
wUlLen	Длина (в пикселах) подчеркивания.
wTextLen	Длина текста пункта меню.

Комментарии:

Структура описывает конкретный пункт меню из списка, ссылка на который содержится в соответствующей структуре MENU.

Версии: 3.0, 3.1.

MODULE

Описание:

```
struct MODULE{
    WORD wSign;
    WORD wLoads;
    struct{
        WORD wFirstOrd;
        WORD wLastOrd;
```

```
    struct{
        BYTE byType;
        BYTE byFlags;
        BYTE bySegNum;
        WORD wOfs;
    }Entry[1];
}NEAR *EntriesTab;
WORD wNext;
struct SGMTAB{
    WORD wOfs;
    WORD wLen;
    WORD wFlags;
    WORD wLenMem;
    WORD wSel;
}NEAR *pDGroup[1];
struct{
    BYTE byLen;
    BYTE byFlag;
    WORD reserved;
    WORD wDate;
    WORD wTime;
    BYTE Name[1];
}NEAR *FileInfo;
WORD wFlags;
WORD wDGroupOrd;
WORD wHeapSize;
WORD wStackSize;
LPVOID lpInit;
LPVOID lpStack;
WORD wSegCount;
WORD wRefEntries;
WORD wStatTab;
SGMTAB NEAR *SegTab;
struct{
    WORD wAlign;
    struct{
        WORD wType;
        WORD wEntries;
        FARPROC lpfnContr;
        struct{
            WORD wOfs;
            WORD wLen;
            WORD wFlags;
            WORD wId;
            HANDLE hSeg;
            WORD wInUse;
        }Resource[1];
    }ResType[1];
}
```

```

    }NEAR *ResTab;
    WORD pResNames;
    WORD pRefTab;
    WORD pImpNames;
    DWORD dwStatTab;
    WORD wMovEntries;
    WORD wSelAlign;
    BOOL bTrueType;
    BYTE byOS;
    BYTE byFlags2;
    WORD pImpNames2;
    WORD reserved;
    WORD reserved2;
    WORD wVer;
};

```

Поля:

wSign	Сигнатура 'NE'.
wLoads	Число загруженных экземпляров модуля. Когда это поле уменьшается до нуля, структура удаляется из памяти.
EntriesTab	Таблица входов для модуля (см. ниже).
wNext	Селектор следующего модуля в списке. Дескриптор первого модуля содержится в поле hFirstModule структуры THHOOK.
pDGroup[1]	Короткая ссылка на первый элемент таблицы структур SGMTAB (описания сегментов) для DGROUP модуля.

Структура SGMTAB:

wOfs	Смещение сегмента в файле, сдвинутое на wSelAlign бит вправо.
wLen	Длина сегмента на диске.
wFlags	Флаги типа памяти сегмента. Значения установленных бит:

- 0 (0x0001) — сегмент данных;
- 4 (0x0010) — перемещаемый;
- 5 (0x0020) — разделяемый;
- 6 (0x0040) — сегмент PRELOAD;
- 7 (0x0080) — только для чтения;

8 (0x0100) — подстраиваемый;
 12 (0x1000) — сбрасываемый.

wLenMem Начальная длина сегмента в памяти (0 соответствует 64K).
 wSel Селектор сегмента.
 FileInfo Ссылка на информационную структуру файла модуля.

Информация о файле:

byLen Длина этой информационной структуры.
 byFlag Бит 1 установлен в версии 3.0, если модуль загружен до Program Manager.
 wDate Дата в формате MS DOS.
 wTime Время в формате MS DOS.
 Name[1] Первый байт ASCIIZ-строки, содержащей имя файла.
 wFlags Информационные флаги модуля.
 Значения отдельных бит:

15 (0x8000) — установлен для DLL;
 11 (0x0800) — приложение с внутренним загрузчиком;
 8 и 9 (0x0300) — программа использует для вывода на экран средства API Windows;
 6 (0x0040) — внутреннее выделение памяти;
 4 (0x0010) — программа использует LIM EMS 3.2;
 1 (0x0002) — каждый экземпляр модуля в памяти имеет собственный DGROUP (обычная задача);
 0 (0x0001) — DGROUP — общий для всех экземпляров (DLL).

wDGroupOrg Номер логического сегмента для DGROUP.
 wHeapSize Начальный размер локального хипа.
 wStackSize Начальный размер стека.
 lpInit Начальный логический указатель для CS:IP.

lpStack	Начальный логический указатель для SS:SP.
wSegCount	Число элементов в SegTab.
wRefEntries	Число элементов в таблице ссылок модуля.
wStatTab	Длина таблицы нерезидентных имен.
SegTab	Таблица структур SEGTab (см. выше), описывающих сегменты модуля.

RESTAB — таблица ресурсов.

Заголовок:

wAlign	Число бит для сдвига влево значений полей Resource[n].wOfs и Resource[n].wLen.
ResType[1]	Первый элемент таблицы типов.

Таблица типов:

wType	Тип ресурсов, таблицу которых содержит данный элемент. Если старший бит установлен, то остальные ($wType \&= 0x7FFF$) содержат одну из констант RT_...
wEntries	Число элементов в таблице ресурсов данного типа.
lpfnContr	Адрес соответствующей функции контроля ресурсов.
Resource[1]	Первый элемент таблицы ресурсов данного типа.

Таблица ресурсов данного типа:

wOfs	Смещение ресурса в файле, сдвинутое на wAlign бит вправо.
wLen	Длина ресурса в файле, сдвинутая на wAlign бит вправо.
wFlags	Флаги типа блока памяти ресурса. Значения отдельных бит:

12 (0x1000) — сбрасываемый;

6 (0x0040) — предварительная загрузка (preload);

	5 (0x0020) — только для чтения;
	4 (0x0010) — перемещаемый;
	2 (0x0004) — загруженный в память.
wId	Идентификатор ресурса.
hSeg	Дескриптор сегмента памяти, содержащей ресурс.
wUse	Счетчик использования ресурса.
pResNames	Короткий указатель на таблицу резидентных имен. Это последовательность паскаль-строк (первый байт содержит длину следующей за ним строки). На первый байт за концом таблицы указывает pRefTab.
pRefTab	Указатель на таблицу ссылок модуля. Таблица содержит последовательность слов, содержащих дескрипторы модулей, логические номера которых являются индексами в этой таблице.
pImpNames	Указатель на нулевой байт перед таблицей импортируемых имен. Формат аналогичен pResNames.
dwStatTab	Смещение в соответствующем NE-файле таблицы нерезидентных имен.
wMovEntries	Число перемещаемых входов.
wSelAlign	Степень двойки, на которую нужно умножить значение поля wOffs соответствующей структуры SEGTab, чтобы получить смещение сегмента в файле. Чаще всего содержит 4 ($wOffs * 16$) или 9 ($wOffs * 512$).
bTrueType	TRUE, если модуль содержит TrueType-шрифты.
byOS	Идентификатор операционной системы, для которой создан модуль: 0 — неизвестная система, 1 — MS OS/2, 2 — Windows, 3 — европейский DOS4 (?), 4 — WIN32.

byFlags2	Дополнительные информационные флаги. Значения отдельных бит:
0	(0x01) — используются длинные имена файлов;
1	(0x02) — задача Windows 2.x, которая может выполняться в защищенном режиме;
2	(0x04) — задача Windows 2.x, поддерживающая векторные шрифты;
3	(0x08) — в файле модуля есть последовательно загружаемые сегменты.
pImpNames2	Соответствует pImpNames, за исключением некоторых экзотических случаев (Windows 3.0 beta).
wVer	Версия Windows, для которой скомпилирован модуль в формате GetVersion().

Комментарии:

Дескриптор модуля (HMODULE), используемый многими функциями Windows, указывает на эту структуру, которая является «идеализированным» вариантом заголовка файла NE.

Версии: 3.0, 3.1.

PALETTEOBJ

Описание:

```
struct PALETTEOBJ {
    GDIOBJHDR Header;
    LOGPALETTE logPalette;
};
```

Поля:

Header	Заголовок объекта (см. GDIOBJHDR).
logPalette	Соответствующая структура LOGPALETTE.

Комментарии:

HPALETTE — дескриптор блока памяти этой структуры в локальном хипе GDI.

Версии: 3.0, 3.1.

PENOBJ

Описание:

```
struct PENOBJ{  
    GDIOBJHDR Header;  
    LOGPEN logPen;  
};
```

Поля:

Header	Заголовок объекта (см. GDIOBJHDR).
logPen	Соответствующая структура LOGPEN.

Комментарии:

HPEN — дескриптор блока памяти этой структуры в локальном хипе GDI.

Версии: 3.0, 3.1.

SELTAB

Описание:

```
struct SELTAB{  
    DWORD dwArenaOffs[1];  
};
```

Поля:

dwArenaOffs[1]	Первый элемент таблицы смещений структур GLOBAL_ARENA.
----------------	--

Комментарии:

Таблица позволяет для любого селектора узнать 32-битное смещение соответствующей структуры GLOBAL_ARENA внутри сегмента BURGERMASTER. Значение селектора с тремя замаскированными младшими битами, деленное на два, является индексом в данной таблице.

Таблица расположена в сегменте BURGERMASTER. Ее размер и смещение в сегменте в версии 3.1 можно получить соответственно из полей wSelTabLen и dwSelTabStart структуры THHOOK. В Windows 3.0 эти переменные расположены в сегменте данных KERNEL со смещениями соответственно 324h и 326h.

Версии: 3.0, 3.1, только расширенный (386) режим.

TASK_QUEUE

Описание:

```
struct TASK_QUEUE{
    WORD wNext;
    HANDLE hOwner;
    WORD wMsgSize;
    WORD wWaitMsgs;
    WORD wNextMsg;
    WORD wNextAvail;
    WORD wQueueEnd;
    DWORD dwTime;
    DWORD dwPos;
    WORD reserved;
#ifdef WIN31
    DWORD dwExtra;
#endif
    DWORD reserved2;
    DWORD dwLParam;
    WORD wParam;
    WORD wMsg;
    HANDLE hWnd;
    BOOL bPostQMsg;
    WORD wExitCode;
#ifdef WIN30
    HANDLE hInst;
#endif
}
```

```

        HANDLE hInSend;
        HANDLE hNextReply;
        HANDLE hNextServe;
#ifdef WIN30
        WORD wQueueState;
        WORD wQueueFlags;
#else
        WORD wQueueFlags;
        WORD wQueueState;
#endif
        BYTE Msgs[1];
};

```

Поля:

wNext	Селектор следующей структуры TASK_QUEUE в общем списке.
hOwner	Дескриптор задачи-владельца данной структуры.
wMsgSize	Размер одного сообщения в данной очереди.
wWaitMsgs	Число сообщений в очереди, ожидающих обработки.
wNextMsg	Смещение первого в очереди сообщения для обработки от начала сегмента очереди.
wNextAvail	Смещение ближайшей доступной ячейки для помещения нового сообщения в очередь.
wQueueEnd	Смещение первого байта за концом очереди сообщений.
dwTime	Значение, возвращаемое функцией GetMessageTime().
dwPos	Значение, возвращаемое функцией GetMessagePos().
dwExtra	Значение, возвращаемое функцией GetMessageExtraInfo().
dwLParam	LParam последнего вызова SendMessage().
wParam	wParam последнего вызова SendMessage().

wMsg	Последнее посланное SendMessage() сообщение.
hWnd	Окно, которому было послано последнее сообщение функцией SendMessage().
bPostQMsg	TRUE после посылки PostQuitMessage().
wExitCode	Код возврата PostQuitMessage().
hInst	Дескриптор экземпляра для задачи.
hInSend	Дескриптор структуры TASK_QUEUE задачи, вызвавшей InSendMessage().
hNextReply	Дескриптор структуры TASK_QUEUE последней задачи, пославшей сообщение текущей задаче функцией SendMessage(). Пославшая сообщение задача после обработки восстанавливает предыдущее значение этого поля.
hNextServe	Предыдущее значение поля hNextReply структуры TASK_QUEUE задачи, которой текущей задачей было послано сообщение функцией SendMessage(). Таким образом, при серии вызовов этой функции образуется очередь LIFO ¹ структур TASK_QUEUE, связанных через это и предыдущее поле.
wQueueState	Комбинация флагов QS_..., описывающих текущее состояние очереди. В Windows 3.0 эти флаги не документированы, и их значения отличаются от используемых в версии 3.1. Флаг QS_MOUSE там имеет значение 1, QS_KEY — 2, QS_POSTMESSAGE — 4, QS_TIMER — 8, QS_PAINT — 0x10 и QS_SENDMESSAGE — 0x40.

¹ Last in - first out — последний пришел — первый ушел.

wQueueFlags	Поле соответствует wQueueState, но не сбрасывается в 0 после вызова функций GetQueueStatus() или GetInputStste() для задачи.
Msgs[1]	Начало буфера для сообщений. По умолчанию буфер может содержать 8 сообщений для локальной и 120 для системной очереди. Однако эти значения могут быть изменены соответственно ключами <i>DefaultQueueSize=</i> и <i>TypeAhead=</i> секции <i>[Windows]</i> файла Win.ini. Поле является таблицей (точнее, кольцевым буфером) структур соответственно MSG для локальной и EVENTMSG для системной очереди.

Комментарии:

Структура описывает организацию очередей сообщений Windows. Как следует из структуры, система посылает сообщения не окнам, а задачам, которые уже занимаются их распределением.

Система содержит одну специальную системную очередь сообщений, не входящую в общий список и не принадлежащую никаким задачам (поля wNext и hOwner не определены), в которую помещаются аппаратные сообщения от внешних устройств и к которой обращаются все задачи.

Очередь располагается в дополнительном (не DGROUP) сегменте данных USER со смещением 2 (версия 3.0) или 0 (3.1).

Версии: 3.0, 3.1.

TDB

Описание:

```
struct TDB{  
    WORD wNext;  
    LPVOID lpStack;  
    WORD wSysMsgs;
```

```

WORD wPriority;
WORD reserved;
WORD wTask;
WORD reserved2[3];
WORD wFldcw;
WORD wType;
WORD reserved3;
WORD wVer;
HANDLE hInst;
HANDLE hModule;
WORD wTaskQueue;
WORD wParent;
WORD reserved4;
LPPROC lpfnSidHandler;
LPPROC reserved5[2];
LPPROC lpfnInt0;
LPPROC lpfnInt2;
LPPROC lpfnInt4;
LPPROC lpfnInt6;
LPPROC lpfnInt7;
LPPROC lpfnInt3Eh;
LPPROC lpfnInt75h;
DWORD dwCompat;
WORD reserved6[7];
WORD wPDB;
LPVOID lpDTA;
BYTE byDrive;
BYTE Path[43];
HANDLE hDirected;
WORD reserved7[2];
WORD wCode;
THUNKS ThnkFirst;
BYTE Name[8];
WORD wSign;
BYTE PDB[256];

```

```
};
```

Поля:

wNext

Селектор следующей структуры TDB в системном списке, NULL для последней задачи. Селектор первой структуры списка возвращается в старшем слове DWORD возврата функции GetCurrentTask().

lpStack

Текущие SS:SP задачи.

wSysMsgs	Текущее число необработанных сообщений в <i>системной</i> очереди, касающихся данной задачи.
wPriority	Приоритет данной задачи. Значения находятся в интервале -32...15. Минимальное значение соответствует максимальному приоритету (список задач сортируется по этому полю), причем значения, бóльшие нуля, практически блокируют задачу.
wTask	Селектор данной структуры. Может использоваться для контроля.
wFldcw	Текущее слово состояния сопроцессора для задачи.
wType	Специальное значение типа задачи: 1 для DOS, 8 для OS/2 1.x и 16 для Win32s под Windows 3.1.
wVer	Версия Windows, для которой скомпилирована задача.
hInst	Дескриптор экземпляра для задачи.
hModule	Дескриптор модуля задачи.
hTaskQueue	Селектор сегмента соответствующей структуры TASK_QUEUE.
wParent	Дескриптор TDB задачи, запустившей данную.
lpfnSigHandler	Адрес обработчика Ctrl-Break задачи.
lpfnInt*	Адреса обработчиков соответствующих прерываний для данной задачи.
dwCompat	«Флаги совместимости» задачи.
wPDB	Селектор PDB задачи.
lpDTA	Адрес DTA (Disk Transfer Area) задачи.
byDrive	Текущий номер дисководов с установленным старшим битом (A=0x80).
Path	Текущий каталог для задачи.
hDirected	Дескриптор задачи, которой было передано управление функцией DirectedYield().

wCode	Дубликат селектора сегмента кода задачи.
ThnkFirst	Первый (часто единственный) элемент таблицы точек входа экспортируемых процедур задачи, созданных функцией MakeProcInstance().
Name	Имя модуля (обычно соответствует имени исполняемого файла).
wSign	Сигнатура 'TD'.
PDB	PDB (PSP защищенного режима) задачи.

Комментарии:

Дескриптор задачи (HTASK) указывает на эту глобальную структуру. Каждый экземпляр задачи имеет собственную структуру TDB.

Структура содержит жизненно важную для системы информацию и вряд ли сильно изменится в будущих версиях.

При использовании задачи рекомендуется проверять равенство количества задач в списке и значения, возвращаемого функцией GetNumTasks().

Версии: 3.0, 3.1.

THUNKS

Описание:

```
struct THUNKS{
    WORD wNext;
    WORD wSign;
    WORD reserved;
    WORD wAvail;
    BYTE Thunks[56];
};
```

Поля:

wNext	Адрес следующей структуры THUNKS в списке, первый элемент которого
-------	--

	содержится в соответствующей структуре TDB.
wSign	Сигнатура 'PT'.
wAvail	Смещение от начала поля wSign ближайшей доступной ячейки для новой точки входа экспортируемой процедуры. Если поле содержит 0, то ячейку следует искать дальше по списку структур.
Thunks	Буфер для семи восьмибитных фрагментов кода — точек входа в процедуры, возвращаемых функцией MakeProcInstance().

Комментарии:

Структура описывает элемент определенного для каждой задачи списка «контейнеров» для точек входа в экспортируемые процедуры, создаваемых функцией MakeProcInstance().

Формат процедуры входа описан в документации и предназначен для помещения в регистр DS селектора сегмента данных по умолчанию программы. Это продиктовано тем, что каждый экземпляр задачи (в отличие от DLL) имеет свой собственный сегмент данных, но код для всех экземпляров общий, и поэтому оказывается сложным использование экспортируемых процедур без динамического создания специальных точек входа.

Решение этой проблемы реализовано компилятором Borland C++ введением установки *Smart Callbacks*, позволяющей не вызывать MakeProcInstance(). Используется тот факт, что регистр SS при входе в любую процедуру Windows уже содержит адрес сегмента данных программы по умолчанию. Таким образом, установка при входе DS равным SS решает проблему.

Версии: 3.0, 3.1.

THHOOK

Описание:

```
struct THHOOK{  
    HANDLE hBM;  
    WORD wBM;
```

```

    HANDLE hFirstMod;
    WORD reserved;
    WORD wTopPDB;
    WORD wFirstPDB;
    WORD reserved2;
    HANDLE hFirstTask;
    HANDLE hCurTask;
    WORD reserved3;
    HANDLE hLock;
#ifdef WIN31
    WORD wSelTabLen;
    DWORD dwSelTabStart;
    BYTE reserved4[220];
#else
    BYTE reserved4[357];
#endif
    FARPROC lpfnCdrNotify;
    HANDLE hTaskFileCdr;
};

```

Поля:

hBM	Дескриптор BURGERMASTER.	структуры
wBM	Селектор сегмента BURGERMASTER.	структуры
hFirstMod	Дескриптор первого модуля в системном списке.	
wTopPDB	PDB текущей задачи.	
wFirstPDB	Дескриптор первой PDB в системе.	
hFirstTask	Дескриптор первой задачи в системном списке.	
hCurTask	Дескриптор текущей задачи.	
hLock	Дескриптор заблокированной в системе задачи или 0, если такой нет.	
wSelTabLen	Длина системной структуры SELTAB.	
dwSelTabStart	Смещение начала структуры SELTAB внутри сегмента BURGERMASTER.	
lpfnCdrNotify	Адрес функции оповещения об изменениях в файловой системе.	
hTaskFileCdr	Задача, вызвавшая функцию FileCdr().	

Комментарии:

Функция содержит важную информацию об организации системы. Ее адрес в Windows 3.1 содержит экспортируемая переменная TНHook. В версии 3.0 адрес структуры не экспортировался, но она была доступна в сегменте данных по умолчанию KERNEL по смещению 0x10.

Версии: 3.0, 3.1.

WND

Описание:

```
struct WND{
    HWND hwndNext;
    HWND hwndChild;
#ifdef WIN31
    HWND hwndParent;
    HWND hwndOwner;
    RECT rectWindow;
    RECT rectClient;
    HANDLE hQueue;
    HRGN hrgnUpdate;
#endif
    HANDLE hClass;
#ifdef WIN31
    HANDLE hInstance;
    FARPROC lpfnWndProc;
    DWORD reserved;
    DWORD dwStyle;
    DWORD dwExStyle;
    HANDLE hControl;
    HANDLE reserved2[3];
#else
    HANDLE reserved;
#endif
    HWND hwndActive;
#ifdef WIN31
    HMENU hSysMenu;
#else
    HANDLE reserved2;
    HANDLE hQueue;
    HRGN hrgnUpdate;
```

```

        HANDLE hDCE;
        HMENU hSysMenu;
        HWND hwndOwner;
        RECT rectWindow;
        RECT rectClient;
        HPALETTE hPal;
        WORD reserved3;
        DWORD dwExStyle;
        DWORD dwStyle;
        HANDLE hControl;
        HANDLE reserved4;
        HWND hwndParent;
        HANDLE hInstance;
        FARPROC lpfnWndProc;
    #endif
    WORD wExtra[1];
};

```

Поля:

hwndNext	Следующее окно в цепочке окон данного иерархического уровня.
hwndChild	Первое окно в списке дочерних окон.
hwndParent	Дескриптор родительского окна.
hwndOwner	Дескриптор окна-владельца.
rectWindow	Положение окна в экранных координатах.
rectClient	Положение рабочей области окна в экранных координатах.
hQueue	Глобальный дескриптор структуры TASK_QUEUE задачи, которой принадлежит окно.
hrgnUpdate	Суммарная область обновления для окна.
hClass	Дескриптор структуры CLASS, соответствующей данному окну.
hInstance	Экземпляр модуля, которому принадлежит окно.
lpfnWndProc	Адрес оконной процедуры.
dwStyle	Основные флаги стиля окна (WS_...).
dwExStyle	Дополнительные флаги стиля окна (WS_EX...). Стиль WS_EX_TOPMOST (0x8L), позволяющий окну в любом

	состоянии находиться поверх других окон, не имеющих этого стиля, документированный в версии 3.1, может использоваться и под Windows 3.0. Бит 3 этого поля должен быть установлен, чтобы окно могло получать сообщение WM_BEGINGRAB.
hControl	Для управляющих окон (кнопок, списков и т.д.) — их идентификатор, для обычного окна — дескриптор меню или NULL в случае его отсутствия.
hwndActive	Значение, возвращаемое для окна функцией GetLastActivePopup().
hSysMenu	Дескриптор копии или (для клиентов MDI) «оригинала» системного меню окна.
hPal	Дескриптор логической палитры для использования с окном или NULL.
wExtra	Начало области дополнительных данных для окна.

Комментарии:

HWND — дескриптор этой структуры в локальном хипе USER.

Версии: 3.0, 3.1.

Часть II

Сообщения Windows

WM_ACTIVATESHELLWINDOW

Значение:

0x44

Комментарии:

Это сообщение посылается для уведомления функций-наблюдателей при активизации окна оболочки (Shell). Windows регистрирует функцией RegisterWindowsMessage() и часто использует вместо данного сообщение с именем «ACTIVATESHELLWINDOW»¹.

Версия: 3.1.

WM_ALTTABACTIVE

Значение:

0x29

Комментарии:

Посылается некоторым типам диалоговых окон при попытке передачи управления другой задаче нажатием на *Alt-Tab*.

Версии: 3.0, 3.1.

¹ Число, возвращаемое функцией RegisterWindowsMessage(), на самом деле является дескриптором глобального атома. Таким образом, получив сообщение со значением в соответствующем диапазоне (от 0xc000 до 0xffff), можно, вызвав функцию GetAtomName(), узнать его имя.

WM_BEGINDRAG

Значение:

0x22c

Комментарии:

Посылается списком (List box) родительским окнам с установленным битом 3 в расширенном описании стиля, когда пользователь перемещает «подсветку».

Обратите внимание, что бит 3 стиля может быть установлен только непосредственно в поле dwExStyle соответствующей структуры WND.

Версии: 3.0, 3.1.

WM_DRAGLOOP

Значение:

0x22d

Параметры:

wParam	1, если окно, над которым находится курсор, может принять перемещаемый объект, 0 в противном случае.
lParam	Дальний указатель на структуру DRAGINFO, содержащую информацию об операции.

Комментарии:

Сообщение посылается функцией DragObject() вызвавшему ее серверу Drag&Drop, чтобы информировать его о процессе перемещения объекта.

Версии: 3.0, 3.1.

WM_DRAGMOVE

Значение:

0x22f

Параметры:

lParam Дальний указатель на структуру DRAGINFO, содержащую информацию об операции.

Комментарии:

Сообщение посылается функцией DragObject() зарегистрированному клиенту Drag&Drop, пока курсор перемещается над ним, и кнопка мыши не отпущена.

Версии: 3.0, 3.1.

WM_DRAGSELECT

Значение:

0x22e

Параметры:

wParam 0, если курсор покидает окно, 1, если входит обратно.
lParam В старшем слове — дескриптор структуры DRAGINFO, содержащей информацию об операции.

Комментарии:

Сообщение посылается функцией DragObject() клиенту Drag&Drop, сообщившему ей, что он может принять объект, когда курсор покидает видимую область окна или попадает в нее.

Версии: 3.0, 3.1.

WM_DROPOBJECT

Значение:

0x22a

Параметры:

wParam	Дескриптор окна сервера Drag&Drop.
lParam	Дальний указатель на структуру DRAGINFO, содержащую информацию об операции.

Комментарии:

Сообщение посылается функцией DragObject() клиенту Drag&Drop, сообщившему ей, что он может принять объект, когда объект «сбрасывается» (Drop) в окно.

Значение, возвращаемое оконной процедурой клиента, полностью соответствует возврату функции DragObject().

Версии: 3.0, 3.1.

WM_ENTERMENULOOP

Значение:

0x211

Комментарии:

Сообщение посылается окну при вызове его системного меню.

Версии: 3.0, 3.1.

WM_ENTERSIZEMOVE

Значение:

0x231

Комментарии:

Сообщение посылается окну, когда начинается его перемещение или изменение размера.

Версии: 3.0, 3.1.

WM_EXITMENULOOP

Значение:

0x212

Комментарии:

Сообщение посылается окну при выходе из его системного меню.

Версии: 3.0, 3.1.

WM_EXITSIZEMOVE

Значение:

0x232

Комментарии:

Сообщение посылается окну, когда заканчивается операция перемещения или изменения размеров окна.

Сообщение не вызывается, если операция производилась с клавиатуры и в результате ни размер, ни позиция окна не изменились.

Версии: 3.0, 3.1.

WM_GETHOTKEY

Значение:

0x33

Комментарии:

Оконная функция после обработки сообщения возвращает слово, младший байт которого содержит код клавиши, а старший — состояние клавиш-переключателей для «горячей клавиши», активизирующей окно, которому послано данное сообщение.

Версии: 3.0, 3.1.

WM_ISACTIVEICON

Значение:

0x35

Комментарии:

Сообщение посылается иконкам — клиентам MDI, чтобы определить их статус. Процедура по умолчанию после обработки возвращает TRUE, если иконка активна, или FALSE, если окно не минимизировано или неактивно.

Версии: 3.0, 3.1.

WM_LBTRACKPOINT

Значение:

0x131

Параметры:

wParam	Номер элемента, выбранного в списке.
lParam	Структура POINT, содержащая координаты курсора мыши относительно левого верхнего угла списка в момент выбора элемента.

Комментарии:

Сообщение посылается родительскому окну окном списка при выборе одного из его элементов.

Версии: 3.0, 3.1.

WM_OTHERWINDOWCREATED

Значение:

0x42

Параметры:

wParam	Дескриптор созданного окна.
--------	-----------------------------

Комментарии:

Это сообщение посылается по цепочке родительских окон, когда создается дочернее окно. Windows регистрирует функцией RegisterWindowsMessage() и часто использует вместо данного сообщение с именем «OTHERWINDOWCREATED».

Версия: 3.1.

WM_OTHERWINDOWDESTROYED

Значение:

0x43

Параметры:

wParam Дескриптор уничтожаемого окна.

Комментарии:

Это сообщение посылается по цепочке родительских окон, когда уничтожается дочернее окно. Windows регистрирует функцией RegisterWindowsMessage() и часто использует вместо данного сообщение с именем «OTHERWINDOWDESTROYED».

Версия: 3.1.

WM_PAINTICON

Значение:

0x26

Комментарии:

Сообщение посылается минимизированным окнам, имеющим иконку класса, когда необходима перерисовка иконки.

Версии: 3.0, 3.1.

WM_QUERYDRPOBJECT

Значение:

0x22b

Параметры:

wParam	0, если курсор находится над рабочей областью окна, иначе 1.
lParam	Дальний указатель на структуру DRAGINFO, содержащую информацию об операции.

Комментарии:

Сообщение посылается функцией DragObject() окну, над которым находится перемещаемый объект, чтобы узнать, может ли оно принять этот объект. В случае возможности приема оконная процедура должна после обработки вернуть TRUE.

Версии: 3.0, 3.1.

WM_SETHOTKEY

Значение:

0x32

Параметры:

wParam	Значение аналогично возврату оконной функции после обработки WM_GETHOTKEY.
--------	--

Комментарии:

Сообщение устанавливает «горячую клавишу» для активизации окна, которому передается это сообщение.

Оконная функция после обработки может возвращать следующие значения: 2, если клавиша уже используется другим окном; 1 — нормальная обработка; 0 — неверный дескриптор окна; -1 — несуществующий код клавиши.

Версии: 3.0, 3.1.

WM_SETVISIBLE

Значение:

0x9

Параметры:

wParam TRUE, если окно становится видимым,
FALSE — если «прячется».

Комментарии:

Функция ShowWindow() посылает окну сообщение перед тем, как спрятать или восстановить его.

Версии: 3.0.

WM_SYSTIMER

Значение:

0x118

Комментарии:

Сообщение полностью аналогично WM_TIMER, но посылается, если таймер установлен без указания процедуры обработки функцией SetSystemTimer().

Версии: 3.0, 3.1.

Часть III

Экспортируемые переменные

__0000H

Описание:

WORD __0000H;

Адрес: KERNEL.183

Комментарии:

__0000H — селектор защищенного режима, указывающий на сегмент физической памяти размером 64К байт с базовым адресом 0000:0000.

Вопреки официальной документации этот селектор (как и любые другие) может быть использован не только в ассемблерной подпрограмме.

Версии: 3.0, 3.1.

__0040H

Описание:

WORD __0040H;

Адрес: KERNEL.193

Комментарии:

__0040H — селектор защищенного режима, указывающий на сегмент физической памяти размером 0x2ff байт по адресу 0040:0000. Это единственный «прозрачный» селектор под Windows:

его значение (в версии 3.1 — значение его внутреннего синонима) также равно 40h.

В любом случае этот селектор надо применять с большой осторожностью — по крайней мере под Windows 3.0 его «поведение» не всегда корректно.

Версии: 3.0, 3.1.

__ROMBIOS

Описание:

WORD __ROMBIOS;

Адрес: *KERNEL.173*

Комментарии:

Это селектор, указывающий на физический сегмент размером 64К по адресу F000:0000.

Версии: 3.0, 3.1.

__WINFLAGS

Описание:

WORD __WINFLAGS;

Адрес: *KERNEL.178*

Комментарии:

Значение переменной аналогично возвращаемому функцией GetWinFlags(), кроме некоторых экзотических случаев с ROM Windows.

Версии: 3.0, 3.1.

fTrapping0

Описание:

BOOL fTrapping0;

Адрес: GDI.355.

Комментарии:

Если переменная установлена в TRUE, то системная ошибка «деление на нуль» обрабатывается внутри GDI, если в FALSE — то обработка прерывания идет по нормальной цепочке.

Версии: 3.0, 3.1.

THHook

Описание:

THOOK FAR *THHook;

Адрес: KERNEL.322

Комментарии:

Переменная содержит адрес системной информационной структуры THHOOK.

Версия: 3.1.

Часть IV

Функции Windows

AllocAlias

Описание:

WORD FAR PASCAL AllocAlias(WORD wSel);

Адрес: KERNEL.172

Аргументы:

wSel Селектор кода в защищенном режиме.

Возвращаемое значение:

Селектор данных, указывающий на тот же сегмент, что и wSel, но доступный и для записи, и для чтения.

Комментарии:

Функция позволяет создавать саомодифицирующийся код в защищенном режиме. Это требует некоторых мер предосторожности. Программа, использующая эту функцию, не должна позволять запускать одновременно несколько своих копий. На все время использования возвращаемого значения сегмент кода wSel должен быть зафиксирован в памяти функцией GlobalPageLock(). После использования селектор должен быть освобожден вызовом FreeSelector().

Версии: 3.0, 3.1.

AllocCStoDSAlias

Описание:

WORD FAR PASCAL AllocCStoDSAlias(WORD wSel);

Адрес: KERNEL.170

Комментарии:

Функция полностью аналогична функции AllocAlias().

Версии: 3.0, 3.1.

AllocSelectorArray

Описание:

WORD FAR PASCAL AllocSelectorArray(WORD wNum);

Адрес: KERNEL.206

Аргументы:

wNum Необходимое число селекторов.

Возвращаемое значение:

Селектор первого дескриптора из таблицы последовательных дескрипторов сегментов.

Комментарии:

Функция избавляет вас от необходимости многократного вызова функции AllocSelector(). Расстояние между селекторами в таблице определяется документированной константой __AHINCR. Каждый селектор должен быть освобожден отдельным вызовом FreeSelector().

Версии: 3.0, 3.1.

BroadcastMessage

Описание:

```
void FAR PASCAL BroadcastMessage(HWND hWnd,  
    WORD wMessage, WORD wParam, DWORD lParam);
```

Адрес: USER.355

Аргументы:

hWnd	Окно, вызывающее функцию.
wMessage	Код сообщения.
wParam, lParam	Дополнительные данные.

Комментарии:

Аналогична вызову PostMessage(0xFFFF,wMessage,wParam,lParam).

Сообщения:

WM_WININICHANGE и WM_DEVMODECHANGE обрабатываются функцией особым образом. Перед посылкой этих сообщений в lParam вместо адреса строки из Win.ini подставляется дескриптор блока памяти, в котором она находится.

Версии: 3.0, 3.1.

CalcChildScroll

Описание:

```
void FAR PASCAL CalcChildScroll(HWND hWnd, WORD  
    wScroll);
```

Адрес: USER.462

Аргументы:

hWnd	Родительское окно MDI.
wScroll	Комбинация флагов SB_HORZ и SB_VERT.

Комментарии:

Инициализирует или убирает полосы прокрутки главного окна приложения MDI, согласуясь с перемещением дочерних окон и изменениями размера материнского окна. Параметр `wScroll` определяет направление (горизонтальное и/или вертикальное) полос, для которых проводится проверка.

Версии: 3.0, 3.1.

CascadeChildWindows

Описание:

```
void FAR PASCAL CascadeChildWindows(HWND hWnd
#ifdef WIN31
    ,WORD wAct
#endif
);
```

Адрес: USER.198

Аргументы:

<code>hWnd</code>	Окно, подокна которого должны быть каскадированы.
<code>wAct</code>	Значение <code>MDITITLE_SKIPDISABLED</code> заставляет функцию не перемещать неактивные окна, любое другое — игнорируется.

Комментарии:

Позволяет каскадировать дочерние окна в не-MDI-приложениях, в том числе и для всей рабочей области окна, вызовом `CascadeChildWindows(GetDesktopWindow(), 0)`.

Обратите внимание, что число аргументов функции зависит от версии Windows. Неправильное задание аргументов приводит к общей ошибке защиты.

Версии: 3.0, 3.1.

ControlPanelInfo

Описание:

```
void FAR PASCAL ControlPanelInfo(int nType,  
    WORD wData, LPSTR lpBuffer);
```

Адрес: USER.273

Аргументы:

nType Команда Панели управления.
wData, lpBuffer Передаваемые или получаемые данные.

Комментарии:

Функция позволяет программам получать или изменять некоторые из параметров, устанавливаемых через Панель управления (Control panel).

В версии Windows 3.1 вместо нее рекомендуется использовать дающую больше возможностей документированную функцию SystemParametersInfo().

Действия функции определяются значениями параметра nType:

1 (CPI_GETBEEP) — если опция «системные звуковые сигналы» неактивна, то lpBuffer устанавливается в NULL, в противном случае — в произвольный ненулевой псевдоуказатель. wData не используется;

2 (CPI_SETBEEP) — ненулевое значение wData включает поддержку звука, ненулевое — отключает. lpBuffer не используется;

3 (CPI_GETMOUSE) — lpBuffer при вызове должен указывать на буфер из трех слов (WORD w1,w2,w3), которые устанавливаются функцией в соответствие с одним из семи возможных значений скорости перемещения мыши ($0 \leq n \leq 6$) следующим образом:

```
if (n==0) w1=w2=w3=0;  
if (n>=1 && n<=3) { w1=13-(3*n); w2=0; w3=1; }  
if (n>=4 && n<=6) { w1=4; w2=24-(3*n); w3=2; }
```

wData не используется;

4 (CPI_SETMOUSE) — Скорость мыши устанавливается в соответствии с содержимым буфера, на который указывает lpBuffer (значения в нем надо установить аналогично CPI_GETMOUSE). wData не используется;

5 (CPI_GETBORDER) — lpBuffer должен быть адресом слова, в которое будет помещено текущее значение толщины рамок окон в пикселах. wData не используется;

6 (CPI_SETBORDER) — wData определяет новую толщину рамок окон в пикселах (от 1 до 50). lpBuffer не используется;

10 (CPI_GETKEYBOARDSPEED) — lpBuffer должен быть адресом слова, в которое будет записано текущее значение скорости повтора для клавиатуры. wData не используется;

11 (CPI_SETKEYBOARDSPEED) — wData определяет новую скорость повтора для клавиатуры (от 1 до 32). lpBuffer не используется;

12 (CPI_LANGDRIVER) — загружает DLL поддержки языка, на ASCIIZ-имя которой указывает lpBuffer. wData не используется;

13 (CPI_ICONSPACING) — если lpBuffer не NULL, то функция считает, что он указывает на слово, в которое надо поместить текущее расстояние между иконками в пикселах. В противном случае из wData берется новое значение для этого расстояния.

Версии: 3.0, 3.1.

CreateSystemTimer

Описание:

```
WORD CreateSystemTimer(WORD wRate,  
                        FARPROC lpfnCallback);
```

Адрес: SYSTEM.2

Аргументы:

wRate	Интервал в миллисекундах, с которым должна вызываться функция таймера. Минимальный возможный интервал для
-------	---

IBM-совместимых машин составляет около 55 мс.

lpfnCallback Дальний указатель на устанавливаемую callback-функцию таймера, которая *должна* находиться в фиксированном (неперемещаемом и несбрасываемом) сегменте.

Возвращаемое значение:

Дескриптор таймера, который будет использоваться при работе с ним и для его освобождения. Если свободных таймеров больше нет (всего в SYSTEM их восемь), то возвращается 0.

Комментарии:

Функция устанавливает асинхронный таймер, другими словами, его процедура (lpfnCallback) вызывается напрямую, в обход очереди сообщений.

Вызов процедуры блокируется при работе полноэкранного нефонового сеанса DOS.

Процедура таймера вызывается непосредственно из обработчика прерывания 8, поэтому рабочие регистры (в том числе DS) там не определены и для доступа к данным программы вам надо перезагружать DS. Многие функции API Windows (кроме простейших типа MessageBeep() или GetCurrentTask()) не могут быть вызваны из обработчика, однако гарантирована возможность вызова оттуда функции PostMessage().

Лучше всего использовать процедуру для абсолютных задержек через изменение значений volatile-переменных, проверяемых в псевдобесконечных циклах.

Версии: 3.0, 3.1.

Death

Описание:

```
void FAR PASCAL Death(HDC hDC);
```

Адрес: GDI.121

Аргументы:

hDC Контекст устройства для всей рабочей поверхности дисплея.

Комментарии:

Функция отключает графический драйвер и переводит экран в текстовый режим по умолчанию (обычно Mode 3 BIOS).

Версии: 3.0, 3.1.

DeletePathName

Описание:

WORD FAR PASCAL DeletePathName(LPSTR lpPathName);

Адрес: KERNEL.76

Аргументы:

lpPathName Полный путь с именем файла.

Возвращаемое значение:

1, если файл не найден или его нельзя удалить.

Комментарии:

Удаляет файл, имя которого содержится в lpPathName.

Версии: 3.0, 3.1.

DiagOutput

Описание:

void FAR PASCAL DiagOutput(LPSTR lpMsg);

Адрес: KERNEL.340

Аргументы:

lpMsg Информационная строка.

Комментарии:

Если Windows были запущены с параметром /B, то строка lpMsg записывается в отладочный файл BOOTLOG.TXT.

Версия: 3.1.

DiagQuery

Описание:

BOOL FAR PASCAL DiagQuery();

Адрес: KERNEL.339

Возвращаемое значение:

TRUE, если Windows были запущены с параметром /B, иначе FALSE.

Версия: 3.1.

DirectedYield

Описание:

void FAR PASCAL DirectedYield(HANDLE);

Адрес: KERNEL.150

Комментарии:

Эта функция документирована в Windows 3.1, однако ей можно пользоваться и в версии 3.0.

Версии: 3.0, 3.1.

DirectResAlloc

Описание:

```
HANDLE FAR PASCAL DirectResAlloc(HANDLE  
    hInstance, WORD wUnknown, WORD wSize);
```

Адрес: KERNEL.168

Аргументы:

hInstance	Экземпляр, вызывающий функцию.
wUnknown	Назначение не известно, должен быть установлен в 0x10.
wSize	Размер необходимого блока.

Возвращаемое значение:

Дескриптор глобального блока памяти, ассоциированного с модулем, экземпляр которого описывает hInstance.

Комментарии:

Созданный функцией блок памяти может быть использован для создания ресурсов «на лету», путем копирования данных в этот разделяемый блок.

Функция полностью аналогична вызову функции *GlobalAlloc(GMEM_DDESHARE, DWORD(wSize))*.

Версии: 3.0, 3.1.

DisableDOS

Описание:

```
void FAR PASCAL DisableDOS();
```

Адрес: KERNEL.42

Комментарии:

Используется при выходе из Windows. Восстанавливает вектора прерываний, переустановленные Windows, чем делает невозможным обращение к диску, запуск новых программ и другие операции, связанные с обращением к DOS.

Версия: 3.0.

DisableKernel

Описание:

```
void DisableKernel();
```

Адрес: KERNEL.125

Комментарии:

Полностью аналогична DisableDOS(), но в отличие от нее не имеет симметричной функции восстановления.

Версии: 3.0, 3.1.

DisableOEMLayer

Описание:

```
void FAR PASCAL DisableOEMLayer();
```

Адрес: USER.4

Комментарии:

Отключает все драйверы ввода-вывода, таймера и сети и восстанавливает исходный режим экрана, стирая его содержимое. Практически программа оказывается на уровне DOS, хотя и остается в защищенном режиме.

Версии: 3.0, 3.1.

DisableSystemTimers

Описание:

```
void FAR PASCAL DisableSystemTimers();
```

Адрес: SYSTEM.5

Комментарии:

Восстанавливает старый вектор INT 8, что заставляет «остановиться» все системные часы.

Версии: 3.0, 3.1.

DOS3Call

Описание:

```
void FAR DOS3Call();
```

Адрес: KERNEL.125

Комментарии:

Функция документирована, но в документации указано, что она может быть вызвана только из ассемблерной процедуры, что, вообще говоря, неверно.

Версии: 3.0, 3.1.

DoSignal

Описание:

```
void FAR PASCAL DoSignal();
```

Адрес: KERNEL.139

Комментарии:

Windows-эквивалент прерывания 1Bh, который непосредственно вызывает обработчик Ctrl-Break окна, имеющего фокус ввода.

Версии: 3.0, 3.1.

DragDetect

Описание:

```
BOOL FAR PASCAL DragDetect (HWND hWnd,  
    LPPOINT lpPoint);
```

Адрес: USER.465

Аргументы:

hWnd	Дескриптор вызывающего окна.
lpPoint	Точка, в которой проверяется начало перемещения (Drag).

Возвращаемое значение:

TRUE, если началось перемещение, FALSE в противном случае.

Комментарии:

Функция может быть вызвана из обработчика WM_LBUTTONDOWN для определения начала перемещения объекта из точки.

Версии: 3.0, 3.1.

DragObject

Описание:

```
DWORD FAR PASCAL DragObject(HWND hwndScope,  
    HWND hwndObj, WORD wType, WORD wOfstruct,  
    PSTR szList, HANDLE hDragCursor);
```

Адрес: USER.464

Аргументы:

hwndScope	Окно, в рамках которого возможно перемещение объекта. Для снятия всех ограничений это должен быть результат GetDesktopWindow().
hwndObj	Окно, из которого производится перемещение.
wType	Тип перемещаемого объекта (см. поле wFlags структуры DRAGINFO).
hOfstruct	Дескриптор глобальной структуры OFSTRUCT.
szList	Список имен перемещаемых объектов, разделенных пробелами.
hDragCursor	Дескриптор курсора <i>или иконки, служащей курсором</i> при перемещении. Любая иконка автоматически преобразуется в черно-белую.

Возвращаемое значение:

Для версии 3.0: 1, если перемещение прошло успешно; 0, если файл не принят клиентом.

Для версии 3.1: 0x544e5250L, если объект принят менеджером печати; 0x454c4946L, если объект принят другой программой; 1, если объект не принят клиентом.

Комментарии:

Функция полностью реализует серверную часть протокола Drag & Drop. Обычно она вызывается из обработчика WM_LBUTTONDOWN, после того как начало перемещения определено функцией DragDetect(). Функция берет на себя

посылку нужных сообщений соответствующим окнам и контроль курсора во время перемещения.

Надо отметить, что File Manager при работе в качестве сервера Drag&Drop помещает в szList имена файлов (но не каталогов), не указывая полного пути к ним. Для получения имени каталога, из которого перемещаются файлы, клиенты должны использовать или функцию GetWindowText() для окна сервера, дескриптор которого передается в структуре DRAGINFO, или (если программа пишется для Windows 3.1) документированную функцию DragQueryFile(). Если серверу необходимо сохранить совместимость с File Manager и со стандартным протоколом перемещения, он должен использовать эти же соглашения и форматы заголовков.

Версии: 3.0, 3.1.

DumpIcon

Описание:

```
DWORD FAR PASCAL DumpIcon(CURSORICONINFO FAR *
    lpInfo, WORD FAR * lpLen, LPSTR FAR *
    lpXORBits, LPSTR FAR * lpANDBits);
```

Адрес: USER.459

Аргументы:

lpInfo	Указатель на информационную структуру для иконки, дамп которой нужно получить.
lpLen	Указатель на слово, содержащее длину структуры CURSORICONINFO.
lpXORBits	Адрес двойного слова, в которое функция записывает указатель на битовую маску XOR иконки.
lpANDBits	Адрес двойного слова, в которое функция записывает указатель на битовую маску AND иконки.

Возвращаемое значение:

0 в случае неуспеха, иначе в младшем слове — размер в байтах одной битовой плоскости, в старшем — размер в байтах всей карты.

Комментарии:

Функция дает возможность получить дамп иконки (в том числе и курсора), информация о которой содержится в lpInfo.

Версии: 3.0,3.1.

EnableDOS

Описание:

```
void FAR PASCAL EnableDOS();
```

Адрес: KERNEL.41

Комментарии:

Функция устанавливает (или восстанавливает, аннулируя действие DisableDOS()) вектора некоторых прерываний на внутренние обработчики Windows, что, в частности, разрешает Windows-программам обращение через DOS к диску и запуск новых процессов.

Версия: 3.0.

EnableKernel

Описание:

```
void FAR PASCAL EnableKernel();
```

Адрес: KERNEL.124

Комментарии:

Функция полностью аналогична EnableDOS().

Версия: 3.0.

EnableOEMLayer

Описание:

```
void FAR PASCAL EnableOEMLayer();
```

Адрес: USER.3

Комментарии:

Восстанавливает все драйверы и менеджер сообщений, замороженные функцией DisableOEMLayer(), а также восстанавливает стандартный графический режим и перерисовывает всю рабочую поверхность.

Версии: 3.0, 3.1.

EnableSystemTimers

Описание:

```
void EnableSystemTimers();
```

Адрес: SYSTEM.4

Комментарии:

Устанавливает на внутренний обработчик Windows вектор прерывания 8, чем запускает системные часы Windows и все связанные с ними логические таймеры.

Версии: 3.0, 3.1.

EndMenu

Описание:

```
void FAR PASCAL EndMenu();
```

Адрес: USER.187

Комментарии:

Если в момент вызова функции было активно меню одного из окон, то это меню принудительно «закрывается», при этом сообщение WM_COMMAND не посылается.

Версии: 3.0, 3.1.

ExcludeVisRect

Описание:

```
int FAR PASCAL ExcludeVisRect(HDC hDC, LPRECT  
    lpRect);
```

Адрес: GDI.73

Аргументы:

hDC	Рабочий контекст устройства.
lpRect	Исключаемый прямоугольник.

Возвращаемое значение:

Тип полученной области.

Комментарии:

Исключает из видимой области (зоны, жестко ограничивающей возможность вывода на DC) прямоугольник lpRect путем непосредственного изменения поля hVisRgn соответствующей структуры DC.

Версии: 3.0, 3.1.

ExitKernel

Описание:

```
void FAR PASCAL ExitKernel(WORD wAction,  
    WORD wRetCode);
```

Адрес: KERNEL.2

Аргументы:

wAction	Определяет действия, выполняемые функцией.
wRetCode	Код возврата из Windows.

Комментарии:

Функция используется для немедленного безусловного выхода из Windows и (возможно) их перезапуска.

Если wAction равно нулю, то происходит выход в DOS со значением возврата wRetCode; при значении 0x42 происходит перезапуск Windows, 0x43 — перезагрузка машины.

Версии: 3.0, 3.1.

FarGetOwner

Описание:

```
WORD FAR PASCAL FarGetOwner(HANDLE hMem);
```

Адрес: KERNEL.404

Аргументы:

hMem	Дескриптор глобального блока памяти.
------	--------------------------------------

Возвращаемое значение:

Если блок разделяемый — дескриптор модуля владельца, иначе — его PDB.

Комментарии:

Функция позволяет определять, какой процесс является владельцем данного блока глобальной памяти.

Имя этой функции *не экспортируется*, подключать ее надо только по адресу в модуле.

Версия: 3.1.

FarSetOwner

Описание:

```
void FAR PASCAL FarSetOwner(HANDLE hMem,  
    WORD wOwnerPDB);
```

Адрес: KERNEL.403

Аргументы:

hMem	Дескриптор глобального блока памяти.
wOwnerPDB	PDB нового владельца блока.

Комментарии:

Функция позволяет изменить владельца глобального блока памяти. Обратите внимание, что если этот блок явно не освобожден, то он освобождается только после завершения как исходного, так и нового процесса-владельца.

Имя этой функции *не экспортируется*, подключать ее надо только по адресу в модуле.

Версия: 3.1.

FastWindowFrame

Описание:

```
BOOL FAR PASCAL FastWindowFrame(HDC hDC,  
    LPRECT lpRect, int xWidth, int yWidth,  
    DWORD dwROP);
```

Адрес: GDI.400

Аргументы:

hDC	Рабочий контекст устройства.
lpRect	Прямоугольник, ограничиваемый рамкой на DC.
xWidth	Толщина (в пикселах) вертикальных линий рамки.
yWidth	Толщина (в пикселах) горизонтальных линий рамки.
dwROP	Код растровой отрисовки (одна из констант, используемых в качестве последнего аргумента функций BitBlt() и StretchBlt()).

Возвращаемое значение:

FALSE, если рамка не может быть нарисована, иначе TRUE.

Комментарии:

Функция позволяет быстро отрисовывать на растровых устройствах рамки толщиной больше одного пиксела. Windows использует эту функцию, когда рисует рамки при передвижении или изменении размеров окон.

Версии: 3.0, 3.1.

FileCdr

Описание:

```
DWORD FAR PASCAL FileCdr(FARPROC lpfnNotify);
```

Адрес: KERNEL.130

Аргументы:

`lpfnNotify` Функция, которая будет вызываться при любом изменении в файловой системе.

Возвращаемое значение:

Если установка callback-функции прошла успешно, то младшее слово возвращаемого значения ненулевое.

В версии 3.1, если в качестве аргумента передано `-1L`, возвращается адрес предыдущего обработчика или `NULL`, если такового не оказалось.

Комментарии:

Эта функция используется File Manager для установки функции-наблюдателя, обновляющей информацию о файловой системе в окнах программы.

Установленная callback-функция вызывается из обработчика прерывания `21h` после обработки запросов, связанных с манипуляциями с файлами или каталогами.

Функция позволяет изменять уже установленный обработчик только той же задаче, которая изначально его установила, сохраняя ее `hTask`. Таким образом, если File Manager уже активен, функция не даст возможности установить новый обработчик изменений и вернет в младшем слове `DWORD` возврата нуль.

Однако это препятствие достаточно легко обходимо: как `hTask` установившей обработчик задачи, так и адрес самого обработчика хранятся соответственно в полях `hTaskFileCdr` и `lpfnCdrNotify` структуры `TNHOOK`, которая полностью доступна для записи. Это позволяет не только устанавливая свою функцию-наблюдателя вне зависимости от текущего состояния системы, но и включать свои обработчики в общую цепочку, сохраняя старый адрес `lpfnCdrNotify` и передавая на него управление после завершения обработки текущих изменений.

Прототип функции-обработчика должен быть следующим:

```
void CALLBACK CdrNotify(WORD wAction, LPSTR  
    lpzPath)
```

lpszPath — полный путь к файлу или каталогу, с которым происходят изменения.

wAction — код конкретного действия, на которое отвечает функция. Этот код — содержимое регистра AX при вызове одной из следующих функций DOS:

- 0x3c — создание файла;
- 0x39 — создание каталога;
- 0x3a — удаление каталога;
- 0x41 — удаление файла;
- 0x43 — получение/установка атрибутов файла;
- 0x56 — переименование файла/каталога;
- 0x57 — установка даты/времени файла;
- 0x5a — создать временный файл;
- 0x5b — создать новый файл;
- 0x6c — расширенное открытие файла.

Версии: 3.0, 3.1.

FillWindow

Описание:

```
void FAR PASCAL FillWindow(HWND hwndParent,
    HWND hWnd, HDC hDC, HANDLE hBrush);
```

Адрес: USER.324

Аргументы:

hwndParent	Материнское окно или NULL.
hWnd	Закрашиваемое окно.
hDC	Рабочий контекст окна.
hBrush	Дескриптор кисти для закрашивания или одна из констант CTLCOLOR_...

Комментарии:

Функция закрашивает окно hWnd кистью hBrush или, если hwndParent не равен NULL (и *только* в этом случае), — стандартными цветами элемента, специфицированного константой CTLCOLOR_... .

Версии: 3.0, 3.1.

FlushCachedFileHandle

Описание:

```
void FAR PASCAL FlushCachedFileHandle(  
    HANDLE hModule);
```

Адрес: KERNEL.319

Аргументы:

hModule Модуль, имеющий открытые файлы.

Комментарии:

Функция закрывает все файлы, открытые в hModule и находящиеся во внутреннем кэше KERNEL, и удаляет ссылки на них.

Версии: 3.0, 3.1.

GdiSeeGdiDo

Описание:

```
DWORD FAR PASCAL GdiSeeGdiDo(WORD wAction,  
    WORD wParam1, WORD reserved, WORD  
    wParam2);
```

Адрес: GDI.452

Аргументы:

wAction	Номер запроса к GDI.
wParam1, wParam2	Данные, передаваемые с запросом.

Возвращаемое значение:

Если значение wAction не поддерживается функцией, то возвращается -1, иначе значение зависит от конкретного запроса.

Комментарии:

Функция дает возможность работы с локальным хипом GDI. Допустимы следующие запросы:

1 (GSGD_LOCALALLOC) — GDI выделяет память в своем локальном хипе вызовом *LocalAlloc(wParam1, wParam2)* и возвращает дескриптор блока в младшем слове;

2 (GSGD_LOCALFREE) — GDI освобождает блок памяти в своем локальном хипе вызовом *LocalFree(wParam1)* и возвращает 0 в случае успеха;

3 (GSGD_LOCALCOMPACT) — функция создает непрерывный свободный блок в локальном хипе GDI, размеры которого передаются в wParam2, и возвращает максимально возможный размер такого блока;

0x103 (GSGD_LOCALHEAP) — GDI возвращает в младшем слове дескриптор своего локального хипа, который может быть использован для доступа к внутренним структурам GDI и выделенным запросом GSGD_LOCALALLOC блокам памяти.

Версии: 3.1.

Get80x87SaveSize

Описание:

WORD Get80x87SaveSize();

Адрес: SYSTEM.7

Возвращаемое значение:

Требуемый размер буфера для сохранения состояния сопроцессора или 0 в случае его отсутствия.

Версии: 3.0, 3.1.

GetAppCompatFlags

Описание:

DWORD FAR PASCAL GetAppCompatFlags (HANDLE hTask);

Адрес: KERNEL.354

Аргументы:

hTask	Задача, исполняемый файл которой был скомпилирован для Windows 3.0.
-------	---

Возвращаемое значение:

«Флаги совместимости» для задачи или 0, если флаги не определены или если задача предназначена для версии 3.1.

Комментарии:

«Флаги совместимости» введены в Windows 3.1 для устранения конфликтов с приложениями, написанными для предыдущей версии системы. Их значения устанавливаются в секции [Compatibility] файла Win.ini; они описаны в некоторых изданиях, комментирующих содержимое этого файла.

Версия: 3.1.

GetClipRgn

Описание:

HRGN FAR PASCAL GetClipRgn (HDC hDC);

Адрес: GDI.173

Аргументы:

hDC Рабочий контекст устройства.

Возвращаемое значение:

Текущая область отсечения на DC.

Версии: 3.0, 3.1.

GetCodeHandle

Описание:

DWORD FAR PASCAL GetCodeHandle(LPVOID lpProc);

Адрес: KERNEL.93

Аргументы:

lpProc Вопреки сказанному в документации это может быть не только дальний указатель на экземпляр процедуры, но и двойное слово, созданное операцией *MK_FP(hModule,wSeg)*, где wSeg — логический номер сегмента в модуле hModule.

Возвращаемое значение:

Младшее слово содержит дескриптор соответствующего сегмента (это WORD возврата, описанный в документации), старшее же содержит *селектор* этого сегмента.

Комментарии:

Если логический сегмент, переданный в параметре, не найден в памяти, то он подгружается с диска.

Версии: 3.0, 3.1.

GetControlBrush

Описание:

```
HBRUSH FAR PASCAL GetControlBrush(HWND hWnd, HDC
    hDC, WORD wType);
```

Адрес: USER.326

Аргументы:

hWnd	Окно-владелец управляющего элемента.
hDC	Рабочий контекст устройства.
wType	Тип управляющего элемента — одна из констант CTLCOLOR_... .

Возвращаемое значение:

Кисть, используемая при отрисовке соответствующего типа управляющих элементов.

Версии: 3.0, 3.1.

GetCurLogFont

Описание:

```
HFONT FAR PASCAL GetCurLogFont (HDC hDC);
```

Адрес: GDI.411

Аргументы:

hDC	Рабочий контекст устройства.
-----	------------------------------

Возвращаемое значение:

Текущий выбранный в контексте шрифт.

Версии: 3.0, 3.1.

GetCurrentPDB

Описание:

DWORD FAR PASCAL GetCurrentPDB();

Адрес: KERNEL.37

Возвращаемое значение:

Вопреки документации функция возвращает не WORD, а DWORD, младшее слово которого содержит значение, описанное в документации, а старшее — собственную PDB текущей задачи.

Версии: 3.0, 3.1.

GetCurrentTask

Описание:

DWORD FAR PASCAL GetCurrentTask();

Адрес: KERNEL.36

Возвращаемое значение:

Вопреки документации функция возвращает не WORD, а DWORD, младшее слово которого содержит значение, описанное в документации, а старшее — дескриптор первой структуры TDB в общем списке.

Версии: 3.0, 3.1.

GetDCState

Описание:

HANDLE FAR PASCAL GetDCState(HDC hDC);

Адрес: GDI.179

Аргументы:

hDC Рабочий контекст устройства.

Возвращаемое значение:

Дескриптор структуры DCSAVE, в которой сохраняется текущее состояние DC.

Комментарии:

Функция, вообще говоря, аналогична SaveDC(), но отличается от нее тем, что не включает структуру DCSAVE в общий «стек» и не изменяет поле wDCSaveLevel соответствующей структуры DC. Восстановить состояние контекста в этом случае можно вызовом SetDCState(), используя значение, возвращаемое этой функцией.

Версии: 3.0, 3.1.

GetExePtr

Описание:

WORD FAR PASCAL GetExePtr(HANDLE h);

Адрес: KERNEL.133

Аргументы:

h Дескриптор блока глобальной памяти или экземпляра задачи или селектор.

Возвращаемое значение:

Дескриптор модуля, соответствующего h.

Комментарии:

Если в качестве аргумента передается дескриптор модуля, то он без изменений возвращается функцией.

Большинство функций KERNEL, требующих дескриптор модуля в качестве аргумента, преобразуют его, используя алгоритм

этой функции, и таким образом могут принимать также дескриптор экземпляра (HINSTANCE).

Версии: 3.0, 3.1.

GetExeVersion

Описание:

WORD FAR PASCAL GetExeVersion();

Адрес: KERNEL.105

Возвращаемое значение:

Версия Windows, для которой скомпилирован исполняемый файл текущей задачи. Формат возврата функции такой же, как и у GetVersion().

Версии: 3.0, 3.1.

GetExpWinVer

Описание:

WORD FAR PASCAL GetExpWinVer(HANDLE hModule);

Адрес: KERNEL.167

Аргументы:

hModule Модуль, версия которого выясняется.

Возвращаемое значение:

Минимальная версия Windows, под которой может исполняться модуль, причем версия находится в старшем байте возвращаемого значения, а модификация — в младшем (например, для Windows 3.1 — 0x30a).

Версии: 3.0, 3.1.

GetFilePortName

Описание:

```
WORD FAR PASCAL GetFilePortName(LPOFSTRUCT  
    lpOfstr);
```

Адрес: USER.343

Аргументы:

lpOfstr Указатель на структуру OFSTRUCT, куда функция запишет свой результат.

Возвращаемое значение:

-3, если пользователь отменил запрос (кнопкой «Cancel» или нажатием на Esc); 1 в противном случае.

Комментарии:

Функция выводит на экран стандартное диалоговое окно «печать в файл» и ждет ввода пользователя. Если файл, назначенный для вывода, уже существует, то функция предупреждает об этом пользователя, выводя соответствующее окно сообщения.

Если запрос не отменен, то выбор пользователя возвращается в структуру, на которую указывает lpOfstruct.

Версии: 3.0, 3.1.

GetFreeMemInfo

Описание:

```
DWORD FAR PASCAL GetFreeMemInfo();
```

Адрес: KERNEL.316

Возвращаемое значение:

В младшем слове — число незаблокированных четырехкилобайтных страниц в системе, в старшем — число свободных страниц.

Версия: 3.1, только расширенный (386) режим.

GetHeapSpaces

Описание:

DWORD FAR PASCAL GetHeapSpaces (HANDLE hInst) ;

Адрес: KERNEL.138

Аргументы:

hInst	Дескриптор модуля или экземпляра, информацию о ресурсах которого должна вернуть функция.
-------	--

Возвращаемое значение:

В старшем слове — общий размер в байтах локального хипа hInst, в младшем — размер свободной области хипа.

Комментарии:

Функция не учитывает некоторых деталей распределения ресурсов в Windows 3.1, поэтому в этой версии для определения процента свободных ресурсов системы (процент свободного пространства в локальных хипах GDI и USER) лучше использовать документированную функцию GetFreeSystemResources().

Версии: 3.0, 3.1.

GetIconId

Описание:

```
WORD FAR PASCAL GetIconId(HANDLE hResource,  
    DWORD dwResType);
```

Адрес: USER.455

Аргументы:

hResource	Дескриптор курсора или иконки.
dwResType	Уточнение типа ресурса: 1 для курсора и 3 для иконки.

Возвращаемое значение:

Идентификатор ресурса, соответствующего hResource, в файле.

Версия: 3.1.

GetInternalWindowPos

Описание:

```
WORD FAR PASCAL GetInternalWindowPos(HWND hWnd,  
    LPRECT lpRectWnd, LPPOINT lpPointIcon);
```

Адрес: USER.460

Аргументы:

hWnd	Окно, расположение которого выясняется.
lpRectWnd	Указатель на структуру RECT, в которую функция записывает текущее расположение окна.
lpPointIcon	Указатель на структуру POINT, в которую возвращается положение окна в минимизированном виде.

Возвращаемое значение:

Текущее состояние окна: одна из документированных констант SW_... .

Комментарии:

В версии 3.1 появилась соответствующая документированная функция `GetWindowPlacement()`, однако она оказалась несколько громоздкой и к тому же не совсем корректно работающей в некоторых ситуациях, так что ее использование вместо данной функции не всегда оправдано. Надо отметить, что данная функция присутствует (и по-прежнему не документирована) в Windows NT.

Версии: 3.0, 3.1.

GetLpErrMode

Описание:

`LPBYTE GetLpErrMode();`

Адрес: KERNEL.99

Возвращаемое значение:

Адрес внутренней переменной `KERNEL`, которая устанавливается в 1 при обработке критической ошибки по `int 24h`.

Версии: 3.0, 3.1.

GetModuleHandle

Описание:

`DWORD FAR PASCAL GetModuleHandle(LPVOID
lpModName);`

Адрес: KERNEL.47

Аргументы:

lpModName Либо указатель на имя модуля, либо двойное слово, созданное операцией *MK_FP(0,h)*, что делает вызов функции аналогичным вызову *GetExePtr(h)*.

Возвращаемое значение:

Вопреки документации функция возвращает не WORD, а DWORD, младшее слово которого содержит значение, описанное в документации, а старшее — дескриптор первого модуля в системном списке (обычно это KERNEL).

Версии: 3.0, 3.1.

GetMouseEventProc

Описание:

FARPROC FAR PASCAL GetMouseEventProc();

Адрес: USER.337

Возвращаемое значение:

Адрес асинхронной процедуры, отвечающей на перемещения и нажатия кнопок мыши.

Комментарии:

Процедуру можно использовать для программной имитации действий мыши без непосредственной посылки сообщений.

Информация о манипуляциях с мышью передается функции через регистры: AX содержит комбинацию флагов ME_..., BX — горизонтальное и CX — вертикальное перемещения мыши с момента последнего вызова функции системой. BX и CX определены, если в AX установлен флаг ME_MOVE. DX должен содержать значение 2.

Версии: 3.0, 3.1.

GetNextQueueWindow

Описание:

```
HANDLE FAR PASCAL GetNextQueueWindow(HWND hWnd,  
    BOOL bToTop);
```

Адрес: USER.274

Аргументы:

hWnd	Окно, от которого ведется отсчет.
bToTop	TRUE, если требуется ближайшее сверху окно, FALSE — ближайшее снизу.

Возвращаемое значение:

Дескриптор одного из двух соседних с hWnd в Z-иерархии окон.

Версия: 3.0.

GetPhysicalFontHandle

Описание:

```
HANDLE FAR PASCAL GetPhysicalFontHandle(HDC hDC);
```

Адрес: GDI.352

Аргументы:

hDC	Рабочий контекст устройства.
-----	------------------------------

Возвращаемое значение:

Дескриптор модуля, содержащего текущий выбранный в контексте шрифт.

Версии: 3.0, 3.1.

GetQueueStatus

Описание:

WORD FAR PASCAL GetQueueStatus();

Адрес: USER.334

Возвращаемое значение:

Текущее состояние очереди сообщений. Значение берется из поля wQueueState структуры TASK_QUEUE.

Версии: 3.0, 3.1.

GetRelAbs

Описание:

int FAR PASCAL GetRelAbs(HDC hDC);

Адрес: GDI.86

Аргументы:

hDC Рабочий контекст устройства.

Возвращаемое значение:

1, если используются абсолютные координаты; 2 — относительные; 0 — если hDC равен NULL.

Комментарии:

Функция определяет текущее поведение функций LineTo() и Polyline(): абсолютные координаты отсчитываются от начала координат контекста, относительные — от текущей позиции пера.

Версии: 3.0, 3.1.

GetSelectorBase

Описание:

DWORD FAR PASCAL GetSelectorBase(WORD);

Адрес: KERNEL.196

Комментарии:

Функция документирована только в Windows 3.1, но она существует и может быть вызвана также в версии 3.0.

Версии: 3.0, 3.1.

GetSelectorLimit

Описание:

DWORD FAR PASCAL GetSelectorLimit(WORD);

Адрес: KERNEL.188

Комментарии:

Функция документирована только в Windows 3.1, но она существует и может быть вызвана также в версии 3.0.

Версии: 3.0, 3.1.

GetSetKernelDOSProc

Описание:

FARPROC FAR PASCAL GetSetKernelDOSProc(
FARPROC lpfnDOSProc);

Адрес: KERNEL.311

Аргументы:

lpfnDOSProc Новый адрес для внутреннего обработчика
int 21h KERNEL.

Возвращаемое значение:

Старый адрес обработчика.

Комментарии:

Функция позволяет перехватывать обращения к функциям DOS3Call() и NoHookDosCall(), которые непосредственно не инициируют прерываний.

Версия: 3.1.

GetTaskDS

Описание:

WORD FAR PASCAL GetTaskDS();

Адрес: KERNEL.155

Возвращаемое значение:

Дескриптор сегмента данных по умолчанию для текущей программы.

Комментарии:

Эту функцию имеет смысл использовать только в драйверах и обработчиках прерываний или асинхронных событий (таких как процедура, устанавливаемая функцией CreateSystemTimer()): в обычных программах этот дескриптор содержится в параметре, передаваемом функции WinMain() под именем hInstance.

Версии: 3.0, 3.1.

GetTaskQueue

Описание:

WORD FAR PASCAL GetTaskQueue (HANDLE hTask) ;

Адрес: KERNEL.35

Аргументы:

hTask Дескриптор задачи (0 — текущий).

Возвращаемое значение:

Дескриптор структуры TASK_QUEUE указанной задачи.

Версии: 3.0, 3.1.

GetTaskQueueDS

Описание:

void FAR PASCAL GetTaskQueueDS ();

Адрес: KERNEL.118

Комментарии:

Функция помещает селектор структуры TASK_QUEUE, соответствующей текущей задаче, в регистр DS. После использования значение регистра должно быть восстановлено.

Версии: 3.0, 3.1.

GetTaskQueueES

Описание:

void FAR PASCAL GetTaskQueueES ();

Адрес: KERNEL.119

Комментарии:

Функция аналогична GetTaskQueueDS(), но в отличие от нее помещает результат не в DS, а в ES, как видно из названия.

Версии: 3.0, 3.1.

GetTimerResolution

Описание:

```
DWORD FAR PASCAL GetTimerResolution();
```

Адрес: USER.14

Возвращаемое значение:

Согласно документации — разрешение таймера (в микросекундах между прерываниями).

Комментарии:

Функция документирована в Windows 3.1, но может вызываться и в версии 3.0. Для IBM-совместимых машин функция возвращает 1000, т.е. одно прерывание в миллисекунду, что, мягко говоря, не слишком близко к истине.

Версии: 3.0, 3.1.

GlobalFreeAll

Описание:

```
void FAR PASCAL GlobalFreeAll(WORD wPDB);
```

Адрес: KERNEL.26

Аргументы:

wPDB PDB задачи (0 — текущий).

Комментарии:

Функция освобождает всю память (кроме разделяемых блоков), выделенную задачей в глобальном хипе.

Версии: 3.0, 3.1.

GlobalHandleNoRIP

Описание:

DWORD FAR PASCAL GlobalHandleNoRIP(WORD wMem);

Адрес: KERNEL.159

Аргументы:

wMem Селектор глобальной памяти.

Возвращаемое значение:

Соответствующий wMem дескриптор.

Комментарии:

Функция отличается от GlobalHandle() тем, что в случае передачи некорректного параметра не завершает вызвавшую программу, иницилируя общую системную ошибку, а просто возвращает нуль.

Версии: 3.0, 3.1.

GlobalMasterHandle

Описание:

DWORD FAR PASCAL GlobalMasterHandle();

Адрес: KERNEL.28

Возвращаемое значение:

В старшем слове — селектор структуры BURGERMASTER, в младшем — ее дескриптор.

Версии: 3.0, 3.1.

IconSize

Описание:

```
DWORD FAR PASCAL IconSize();
```

Адрес: USER.86

Возвращаемое значение:

В старшем слове — ширина и в младшем — высота иконок в пикселах на текущем экране.

Версии: 3.0, 3.1.

InitAtomTable

Описание:

```
WORD FAR PASCAL InitAtomTable(int nSize);
```

Адрес: KERNEL.68

Комментарии:

Вопреки документации функция возвращает не просто BOOL, а WORD, который содержит короткий указатель на созданную структуру ATOMTABLE, или 0 в случае неудачи.

Версии: 3.0, 3.1.

InquireVisRgn

Описание:

```
HRGN FAR PASCAL InquireVisRgn(HDC hDC);
```

Адрес: GDI.131

Аргументы:

hDC Рабочий контекст устройства.

Возвращаемое значение:

Текущая видимая область контекста.

Версии: 3.0, 3.1.

IntersectVisRect

Описание:

```
int FAR PASCAL IntersectVisRect(HDC hDC, int  
                  nLeft, int nTop, int nRight, int nBottom);
```

Адрес: GDI.98

Аргументы:

hDC	Рабочий контекст устройства.
nLeft, nTop	Левый верхний угол прямоугольника.
nRight, nBottom	Правый нижний угол прямоугольника.

Возвращаемое значение:

Тип полученной области.

Комментарии:

Функция создает новую видимую область из пересечения старой области и заданного прямоугольника.

Версия: 3.0.

IsBadReadPtr

Описание:

```
BOOL FAR PASCAL IsBadReadPtr(BYTE FAR * lpBuf,  
    WORD wLen);
```

Адрес: KERNEL.334

Аргументы:

lpBuf	Адрес проверяемого буфера.
wLen	Длина буфера.

Возвращаемое значение:

FALSE, если весь буфер доступен для чтения.

Версия: 3.1.

IsBadStringPtr

Описание:

```
BOOL FAR PASCAL IsBadStringPtr(LPSTR lpzBuffer);
```

Адрес: KERNEL.337

Аргументы:

lpzBuf	Адрес проверяемой строки.
--------	---------------------------

Возвращаемое значение:

FALSE, если lpzBuf является полностью доступной для чтения ASCIIZ-строкой. Длина строки не может превышать 64К.

Версия: 3.1.

IsBadWritePtr

Описание:

```
BOOL FAR PASCAL IsBadWritePtr(BYTE FAR * lpBuf,  
                               WORD wLen);
```

Адрес: KERNEL.334

Аргументы:

lpBuf	Адрес проверяемого буфера.
wLen	Длина буфера.

Возвращаемое значение:

FALSE, если весь буфер доступен для записи.

Версии: 3.1.

IsDCCurrentPalette

Описание:

```
BOOL FAR PASCAL IsDCCurrentPalette(HDC hDC);
```

Адрес: GDI.412

Аргументы:

hDC	Проверяемый контекст устройства.
-----	----------------------------------

Возвращаемое значение:

TRUE, если hDC использует ту же палитру по умолчанию, что и рабочая поверхность (фон) экрана.

Версии: 3.0, 3.1.

IsDCDirty

Описание:

```
BOOL FAR PASCAL IsDCDirty(HDC hDC, LPRECT  
    lpInvRect);
```

Адрес: GDI.169

Аргументы:

hDC	Проверяемый контекст устройства.
lpInvRect	Указатель на структуру RECT, в которую функция в случае необходимости помещает координаты прямоугольника на hDC, требующего перерисовки.

Возвращаемое значение:

TRUE, если контекст нуждается в перерисовке.

Комментарии:

Функция и уничтожает информацию о необходимости перерисовки, предполагая, что вызвавшая ее функция возьмет на себя заботу о перерисовке hDC.

Версии: 3.0, 3.1.

IsGDIObject

Описание:

```
BOOL FAR PASCAL IsGDIObject(HANDLE hObject);
```

Адрес: GDI.462

Аргументы:

hObject	Дескриптор проверяемого объекта.
---------	----------------------------------

Возвращаемое значение:

TRUE, если hObject указывает на один из объектов, содержащихся в локальном хипе GDI.

Версия: 3.1.

IsROMFile

Описание:

BOOL FAR PASCAL IsROMFile(HANDLE hFile);

Адрес: KERNEL.326

Аргументы:

hFile Дескриптор проверяемого файла.

Возвращаемое значение:

TRUE, если hFile находится в ROM-памяти.

Комментарии:

Функция работает только в ROM Windows, в обычной системе она всегда возвращает FALSE.

Версия: 3.1.

IsROMModule

Описание:

BOOL FAR PASCAL IsROMModule(HANDLE hModule);

Адрес: KERNEL.323

Аргументы:

hModule Дескриптор проверяемого модуля.

Комментарии:

Функция аналогична IsROMFile().

Версия: 3.1.

IsSharedSelector

Описание:

BOOL FAR PASCAL IsSharedSelector(WORD wSel);

Адрес: KERNEL.345

Аргументы:

wSel Проверяемый селектор.

Возвращаемое значение:

TRUE, если селектор принадлежит DLL.

Версии: 3.0, 3.1.

IsTaskLocked

Описание:

WORD FAR PASCAL IsTaskLocked();

Адрес: KERNEL.122

Возвращаемое значение:

Дескриптор заблокированной в системе задачи либо 0, если такой не существует.

Комментарии:

Функция определяет, является ли текущая задача единственной исполняющейся в системе (заблокированной).

Блокировка может производиться функциями LockCurrentTask() и LockInput().

Версии: 3.0, 3.1.

IsUserIdle

Описание:

```
BOOL FAR PASCAL IsUserIdle();
```

Адрес: USER.59

Возвращаемое значение:

TRUE, когда весь ввод с клавиатуры и мыши к моменту вызова функции уже обработан (пользователь неактивен).

Комментарии:

IsUserIdle() — одна из тех функций, которые могут вызываться из обработчика CreateSystemTimer().

Версии: 3.0, 3.1.

IsValidMetaFile

Описание:

```
BOOL FAR PASCAL IsValidMetaFile(HANDLE  
    hMetaFile);
```

Адрес: GDI.410

Аргументы:

hMetaFile Дескриптор проверяемого метафайла.

Возвращаемое значение:

TRUE, если аргумент — действительный дескриптор метафайла.

Версии: 3.0, 3.1.

IsWinOldAppTask

Описание:

BOOL IsWinOldAppTask (HANDLE hTask);

Адрес: KERNEL.158

Аргументы:

hTask Дескриптор проверяемой задачи.

Возвращаемое значение:

TRUE, если hTask — дескриптор задачи окна DOS.

Версии: 3.0, 3.1.

Keybd_Event

Описание:

void Keybd_Event();

Адрес: USER.289

Комментарии:

Эта функция непосредственно вызывается обработчиком клавиатурного прерывания 9 и занимается отправкой соответствующих сообщений.

Информация о событиях передается через регистры: AX — виртуальный код клавиши, BX — код зоны OEM, BH — расширенный код клавиши.

Функция может быть использована для физической имитации ввода с клавиатуры.

Версия: 3.1.

KillSystemTimer (I)

Описание:

```
WORD KillSystemTimer(WORD wTimer);
```

Адрес: SYSTEM.3

Аргументы:

wTimer Дескриптор таймера. Это значение возвращается функцией CreateSystemTimer().

Возвращаемое значение:

0 в случае успеха, иначе wTimer.

Комментарии:

Функция освобождает соответствующий системный таймер.

В версии Windows 3.0 существует функция USER, экспортируемая под тем же именем. В версии 3.1 она переименована в BEAR182().

Версии: 3.0, 3.1.

KillSystemTimer (II)

Описание:

```
BOOL FAR PASCAL KillSystemTimer(HWND hWindow,  
WORD wTimer);
```

Адрес: USER.182

Аргументы:

hWindow	Окно-владелец таймера.
wTimer	Дескриптор таймера.

Возвращаемое значение:

TRUE в случае успеха.

Комментарии:

Функция уничтожает таймер, созданный функцией SetSystemTimer().

В версии 3.1 функция экспортируется под именем Bear182(), поэтому рекомендуется подключать ее не указывая имени.

Версии: 3.0, 3.1.

LoadCursorIconHandler

Описание:

```
HANDLE FAR PASCAL LoadCursorIconHandler(HANDLE  
    hMem, HANDLE hInstance, WORD wResId);
```

Адрес: USER.336

Аргументы:

hMem	Блок глобальной памяти для ресурса, или NULL.
hInstance	Экземпляр программы, в исполняемом файле которой содержится ресурс.
wResId	Идентификатор ресурса в файле.

Возвращаемое значение:

Дескриптор загруженной иконки или курсора.

Комментарии:

Функция загружает из файла, соответствующего hInstance, 16-цветную битовую карту «старого формата», которая загружается в hMem и, если тип ресурса — курсор, преобразуется в

монохромную. Если hMem установлен в NULL, то функция сама выделяет нужный блок; если размер выделенного блока недостаточен, то функция выделяет дополнительную память для hMem.

Версии: 3.0, 3.1.

LoadDIBCursorHandler

Описание:

```
HANDLE FAR PASCAL LoadDIBCursorHandler(HANDLE  
    hMem, HANDLE hInstance, WORD wResId);
```

Адрес: USER.356

Комментарии:

Функция отличается от LoadCursorIconHandler() только тем, что работает с форматом DIB и не загружает иконки.

Версии: 3.0, 3.1.

LoadDIBIconHandler

Описание:

```
HANDLE FAR PASCAL LoadDIBIconHandler(HANDLE hMem,  
    HANDLE hInstance, WORD wResId);
```

Адрес: USER.357

Комментарии:

Функция аналогична LoadDIBCursorHandler(), но работает с иконками.

Версии: 3.0, 3.1.

LoadIconHandler

Описание:

HICON FAR PASCAL LoadIconHandler(HANDLE
hResource, BOOL bDIB);

Адрес: USER.456

Аргументы:

hResource	Ресурс, содержащий битовую карту.
bDIB	TRUE, если битовая карта представлена в формате DIB.

Возвращаемое значение:

Дескриптор иконки, полученной преобразованием hResource.

Комментарии:

Функция позволяет преобразовывать битовые карты (в том числе задающие изображение иконок в любом формате) в стандартную 16-цветную иконку.

Версии: 3.0, 3.1.

LocalCountFree

Описание:

WORD FAR PASCAL LocalCountFree();

Адрес: KERNEL.161

Возвращаемое значение:

Размер в байтах свободной области в текущем локальном хипе.

Комментарии:

Текущим считается хип, селектор сегмента данных которого загружен в регистр DS на момент вызова функции. Таким образом, функция позволяет определять размер свободного пространства в хипе любой программы, загрузив ее hInstance в регистр DS, хотя при наличии функции GetHeapSpaces() необходимость таких манипуляций сомнительна.

Версии: 3.0, 3.1.

LocalHandlerDelta

Описание:

```
int FAR PASCAL LocalHandlerDelta(int nNewDelta);
```

Адрес: KERNEL.310

Аргументы:

nNewDelta	Новое значение инкремента для таблицы перемещаемых блоков локальной памяти.
-----------	---

Возвращаемое значение:

Старое значение инкремента.

Комментарии:

Функция позволяет изменять значение поля wMovDelta текущей структуры LOCAL_HEAP.

Версия: 3.1.

LocalHeapSize

Описание:

```
WORD FAR PASCAL LocalHeapSize();
```

Адрес: KERNEL.162

Возвращаемое значение:

Размер в байтах текущего локального хипа.

Версии: 3.0, 3.1.

LocalNotify

Описание:

```
FARPROC FAR PASCAL LocalNotify(FARPROC  
    lpfnNotify);
```

Адрес: KERNEL.14

Аргументы:

lpfnNotify	Новый адрес для функции-обработчика сообщений локального хипа.
------------	--

Возвращаемое значение:

Адрес старого обработчика.

Комментарии:

Эта функция — реликт версий 2.x, и сейчас ей трудно придумать применение. Если вас интересуют подробности, обратитесь к документации по SDK 2.x, там эта функция достаточно подробно описана.

Версии: 3.0, 3.1.

LockCurrentTask

Описание:

```
void FAR PASCAL LockCurrentTask(BOOL bNeedLock);
```

Адрес: KERNEL.33

Аргументы:

bNeedLock TRUE, если задача должна быть заблокирована в системе, FALSE, если блокировка должна быть снята.

Комментарии:

Функция позволяет блокировать текущую задачу, то есть делать ее единственной исполняющейся в системе. Другие задачи замораживаются и не получают никаких сообщений.

Если на момент вызова функции в системе есть задачи DOS, исполняемые в фоновом режиме, то эти задачи *не* приостанавливаются, но пользователь теряет возможность переключения на них.

Версии: 3.0, 3.1.

LockMyTask

Описание:

```
void FAR PASCAL LockMyTask(BOOL bNeedLock);
```

Адрес: USER.276

Комментарии:

Функция аналогична LockCurrentTask(), но обрабатывает еще и состояние системной очереди сообщений.

Версии: 3.0, 3.1.

LongPtrAdd

Описание:

```
void FAR PASCAL LongPtrAdd(LPVOID lpPtr, DWORD dwAdd);
```

Адрес: KERNEL.180

Аргументы:

lpPtr	Модифицируемый дальний указатель.
dwAdd	Прибавляемая к адресу величина.

Возвращаемое значение:

Функция изменяет базовый адрес селектора сегмента, в который указывает lpPtr, увеличивая его на dwAdd.

Версии: 3.0, 3.1.

LookupMenuHandle

Описание:

```
HMENU FAR PASCAL LookupMenuHandle(HMENU hMenu,  
    int nId);
```

Адрес: USER.217

Аргументы:

hMenu	Меню верхнего уровня, с которого начинается поиск.
nId	Искомый идентификатор (команда).

Возвращаемое значение:

Дескриптор меню или подменю, в котором найден элемент, соответствующий nId.

Версии: 3.0, 3.1.

MakeObjectPrivate

Описание:

```
BOOL FAR PASCAL MakeObjectPrivate(HANDLE hObject,  
    BOOL bPrivate);
```

Адрес: GDI.463

Аргументы:

hObject	Один из объектов GDI.
bPrivate	TRUE, если объект надо пометить как «собственный», FALSE, если надо снять пометку.

Возвращаемое значение:

TRUE, если объект уже был помечен.

Комментарии:

Функция помечает hObject как «принадлежащий» задаче, устанавливая бит 13 в поле wType его заголовка (см. GDI_OBJHDR), что делает объект «невидимым» для других задач.

Информация о задаче-владельце в объект не записывается, что позволяет предположить, что функция введена только для повышения безопасности работы с объектом на время его использования в конкретном контексте.

Версия: 3.1.

NoHookDOSCall

Описание:

```
void FAR PASCAL NoHookDOSCall()
```

Адрес: KERNEL.101

Комментарии:

Функция практически аналогична DOS3Call(), за исключением того, что она перед обращением к обработчику int 21h не вызывает функцию WriteOutProfiles(), но сохраняет на время обработки регистр DS.

Версии: 3.0, 3.1.

OffsetVisRgn

Описание:

```
int FAR PASCAL OffsetVisRgn(HDC hDC, int nOffsX,  
    int nOffsY);
```

Адрес: GDI.102

Аргументы:

hDC	Рабочий контекст устройства.	
nOffsX	Расстояние, на которое	область
	перемещается по горизонтали.	
nOffsY	Расстояние, на которое	область
	перемещается по вертикали.	

Возвращаемое значение:

Тип перемещенной области.

Комментарии:

Функция перемещает видимую область hDC.

Версии: 3.0, 3.1.

PaintRect

Описание:

```
void FAR PASCAL PaintRect(HWND hwndParent, HWND  
    hwnd, HBRUSH hBrush, LPRECT lpRect);
```

Адрес: USER.325

Аргументы:

hwndParent	Дескриптор родительского окна.
hwnd	Рабочее окно.
hDC	Рабочий контекст устройства.
hBrush	Кисть для закраски.
lpRect	Закрашиваемый прямоугольник.

Комментарии:

Функция закрашивает прямоугольник окна hWnd, на координаты которого в системе hDC указывает lpRect. Кисть для закрашки задает hBrush. Если hwndParent не равно NULL, hBrush может содержать одну из констант CTLCOLOR_..., задающую стандартный цвет управляющего подокна.

Версии: 3.0, 3.1.

PrestoChangoSelector

Описание:

```
WORD FAR PASCAL PrestoChangoSelector(WORD
    wSource, WORD wDest);
```

Адрес: KERNEL.177

Аргументы:

wSource	Исходный селектор.
wDest	Изменяемая копия wSource, созданная AllocSelector().

Возвращаемое значение:

wDest после изменения или 0 в случае неуспеха.

Комментарии:

Функция инвертирует признак «код/данные» wSelDest, что позволяет писать в область кода и «исполнять» данные.

В версии 3.1 появилась функция ChangeSelector(), которая была документирована еще в версии 3.0 (уникальный случай), хотя там она и не экспортировалась. Единственное отличие от нее функции PrestoChangoSelector()— проверка параметров здесь не производится.

Функция описана в онлайн-документации Borland C++ 3.x *вместо* ChangeSelector(), что не совсем корректно.

Версии: 3.0, 3.1.

RCos

Описание:

```
int FAR PASCAL RCos(int nRadius, int nAngle);
```

Адрес: GDI.177

Аргументы:

nRadius	Масштабный коэффициент.
nAngle	Угол в десятых градуса.

Возвращаемое значение:

Округленное до целого точное значение $g \cos \alpha$, где g — nRadius, α — nAngle/10.

Комментарии:

Функция позволяет быстро (в целых числах) выполнять распространенную тригонометрическую операцию.

Версия: 3.0.

RealizeDefaultPalette

Описание:

```
int FAR PASCAL RealizeDefaultPalette(HDC hDC);
```

Адрес: GDI.365

Аргументы:

hDC	Рабочий контекст устройства.
-----	------------------------------

Возвращаемое значение:

Число элементов логической палитры по умолчанию, соответствующих различным элементам системной палитры.

Комментарии:

Функция восстанавливает палитру по умолчанию для hDC и приводит ее в соответствие с системной (реализует).

Версии: 3.0, 3.1.

RegisterPTrace

Описание:

```
void FAR PASCAL RegisterPTrace(FARPROC
    lpfnNotify);
```

Адрес: KERNEL.202

Аргументы:

lpfnNotify Функция-обработчик системных событий.

Комментарии:

Функция устанавливает Callback-процедуру без параметров и возвращаемого значения. Передача NULL в lpfnNotify отменяет обработку соответствующих событий.

Информация о событиях передается функции через регистры, причем код типа события передается в AH. Обработываются следующие коды:

0 — KERNEL требует вывода одного символа отладочной информации, на который указывает DS:DX;

1 — KERNEL ждет от программы ввода символа для обработки ошибки (Abort/Retry/Ignore — a/i/n). Обычная реакция — запрос пользователю. Символ должен быть возвращен в AL;

0xd — текущая задача завершается;

0xe — текущая задача была только что запущена;

0x12 — KERNEL требует вывод строки отладочной информации, адрес которой передается в ES:SI (DS:SI в версии 3.0);

0x50 — определяется (загружается) новый сегмент. BX содержит порядковый номер сегмента в файле модуля, CX — селектор сегмента, DX — дескриптор экземпляра модуля, которому

принадлежит сегмент, SI — флаги памяти сегмента, ES:(E)DI указывает на имя модуля;

0x52 — освобождается один из сегментов. BX содержит освобожденный селектор;

0x59 — загружается задача. CX:BX содержит адрес ее точки входа;

0x62 — задача завершается. SP+6 указывает на код завершения;

0x63 — прерывание программы по Ctrl-Alt-SysReq;

0x64 — загружается DLL. SI содержит дескриптор модуля библиотеки, CX:BX указывает на точку входа;

0x65 — выгружен из памяти последний экземпляр модуля, дескриптор которого содержится в ES.

По возможности вместо данной функции рекомендуется использовать соответствующие средства ToolHelp.

Версии: 3.0, 3.1.

RepaintScreen

Описание:

```
void FAR PASCAL RepaintScreen();
```

Адрес: USER.275

Комментарии:

Функция заставляет перерисовываться все видимые на экране окна (включая минимизированные и рабочую поверхность).

Версии: 3.0, 3.1.

Restore80x87State

Описание:

```
void Restore80x87State(LPVOID lpState);
```

Адрес: SYSTEM.9

Аргументы:

lpState	Указатель на буфер данных о сопроцессоре, размер которого (обычно 94 байта) возвращается функцией Get80x87SaveSize().
---------	---

Комментарии:

Функция восстанавливает состояние сопроцессора, сохраненное функцией Save80x87SaveState().

Версии: 3.0, 3.1.

RestoreVisRgn

Описание:

```
WORD FAR PASCAL RestoreVisRgn(HDC hDC);
```

Адрес: GDI.130

Аргументы:

hDC	Рабочий контекст устройства.
-----	------------------------------

Возвращаемое значение:

Тип восстановленной области.

Комментарии:

Функция восстанавливает из «стека» конфигурацию видимой области hDC, предварительно сохраненную функцией SaveVisRgn().

Версии: 3.0, 3.1.

Resurrection

Описание:

```
void FAR PASCAL Resurrection(HDC hDC, WORD  
    reserv1, WORD reserv2, WORD reserv3, WORD  
    reserv4, WORD reserv5, WORD reserv6);
```

Адрес: GDI.122

Аргументы:

hDC	Контекст устройства рабочей поверхности экрана.
-----	---

Комментарии:

Функция восстанавливает работу графического драйвера Windows, прерванную функцией Death(), и переводит экран в стандартный графический режим.

Версии: 3.0, 3.1.

RSin

Описание:

```
int FAR PASCAL RSin(int nRadius, int nAngle);
```

Адрес: GDI.178

Аргументы:

nRadius	Масштабный коэффициент (гипотенуза).
---------	--------------------------------------

nAngle Угол в десятых градуса.

Возвращаемое значение:

Аналогично RCos(), но вычисляется (как следует из названия) синус угла.

Версия: 3.0.

Save80x87State

Описание:

```
void Save80x87State(LPVOID lpBuffer);
```

Адрес: SYSTEM.8

Аргументы:

lpState Буфер данных сопроцессора.

Комментарии:

Функция сохраняет состояние сопроцессора в буфере, на который указывает lpState и размер которого (обычно 94 байта) можно узнать, вызвав функцию Get80x87SaveSize(). Действие функции аналогично инструкции сопроцессора *fsave*.

Версии: 3.0, 3.1.

SaveVisRgn

Описание:

```
HRGN FAR PASCAL SaveVisRgn(HDC hDC);
```

Адрес: GDI.129

Аргументы:

hDC Рабочий контекст устройства.

Возвращаемое значение:

Сохраненная копия области.

Комментарии:

Функция сохраняет в «стеке» конфигурацию видимой области hDC.

Версии: 3.0, 3.1.

ScrollChildren

Описание:

```
void FAR PASCAL ScrollChildren(HWND hWnd, WORD  
    wMsg, WORD wSB, DWORD reserved);
```

Адрес: USER.463

Аргументы:

hWnd	Материнское окно MDI.
wMsg	Одно из сообщений прокрутки.
wSB	Одна из констант SB_..., описывающая полосу прокрутки, соответствующую wMsg.

Комментарии:

Функция перемещает дочерние окна сервера MDI в ответ на прокрутку, осуществленную пользователем.

Версии: 3.0, 3.1.

SelectorAccessRights

Описание:

```
WORD FAR PASCAL SelectorAccessRights(WORD wSel,  
    BOOL bSet, WORD wAccRights);
```

Адрес: KERNEL.196

Аргументы:

wSel	Селектор, права доступа которого проверяются или изменяются.
bSet	Если TRUE, то права доступа wSel устанавливаются в соответствии с wAccRights, иначе права доступа возвращаются функцией.
wAccRights	Поле игнорируется, если bSet==TRUE, иначе это комбинация битовых флагов, определяющих новые права доступа для wSel. Значения отдельных бит:

- 0 (0x0001) — к сегменту был доступ;
- 1 (0x0002) — установлен для доступных для записи данных или доступного для чтения кода;
- 2 (0x0004) — если установлен этот бит и бит 1, то данные в сегмент записываются от верхней границы к нижней (стек);
- 3 (0x0008) — установлен для селектора сегмента кода;
- 4 (0x0010) — установлен для селектора, не принадлежащего системе;
- 15 (0x8000) — если установлен, сегмент выравнивается на границу параграфа, иначе выравнивания не происходит.

Возвращаемое значение:

Если bSet==FALSE, то формат возвращаемого слова аналогичен формату поля wAccRights.

Комментарии:

Функция позволяет узнавать или изменять системные права доступа любого сегмента.

Следует обратить внимание, что проверка wSel на действительность проводится не всегда, так что при передаче неверного значения селектора результаты не определены.

Версии: 3.0, 3.1.

SelectVisRgn

Описание:

```
int FAR PASCAL SelectVisRgn(HDC hDC, HRGN hRgn);
```

Адрес: GDI.105

Аргументы:

hDC	Рабочий контекст устройства.
hRgn	Выбираемая область на hDC.

Возвращаемое значение:

Тип выбранной области.

Комментарии:

Функция жестко устанавливает видимую область на hDC.

Версии: 3.0, 3.1.

SetDCOrg

Описание:

```
POINT FAR PASCAL SetDCOrg(HDC hDC, WORD wOrgX,  
                           WORD wOrgY);
```

Адрес: GDI.117

Аргументы:

hDC	Изменяемый контекст устройства.
wOrgX, wOrgY	Новое начало координат на hDC.

Возвращаемое значение:

Предыдущее значение начала координат контекста.

Комментарии:

Проверка действительности hDC не проводится, поэтому передача неверного параметра имеет непредсказуемые последствия.

Версии: 3.0, 3.1.

SetDCState

Описание:

```
void FAR PASCAL SetDCState(HDC hDC, HANDLE  
    hDCSave);
```

Адрес: GDI.180

Аргументы:

hDC	Рабочий контекст устройства.
hDCSave	Дескриптор структуры DCSAVE, возвращенный функцией GetDCState().

Комментарии:

Функция позволяет восстанавливать состояние контекста устройства в обход «стека» SaveDC()/RestoreDC().

Версии: 3.0, 3.1.

SetDCStatus

Описание:

```
BOOL FAR PASCAL SetDCStatus(HDC hDC, BOOL bInval,  
    LPRECT lpRect);
```

Адрес: GDI.170

Аргументы:

hDC	Рабочий контекст устройства.
-----	------------------------------

bInval	Если TRUE, то lpRect должен указывать на координаты прямоугольника, который объявляется на hDC «нуждающимся в перерисовке», если же этот указатель равен NULL, то используется прямоугольник от последней операции обновления DC. Если данное поле содержит FALSE, то в буфер, на который указывает lpRect, функция помещает координаты текущего прямоугольника для перерисовки, после чего система считает, что DC в обновлении больше не нуждается.
lpRect	Адрес структуры RECT для прямоугольника обновления.

Возвращаемое значение:

TRUE, если DC нуждался в обновлении на момент вызова функции.

Версии: 3.0, 3.1.

SetDeskPattern

Описание:

BOOL FAR PASCAL SetDeskPattern();

Адрес: USER.279

Возвращаемое значение:

TRUE в случае успеха.

Комментарии:

Если в файле Win.ini определена штриховка для фона рабочей области окна по умолчанию, то она устанавливается в качестве текущей.

В Windows 3.1 функция экспортируется под именем OldSetDeskPattern(), и все ее возможности реализуются функцией SystemParametersInfo().

Версии: 3.0, 3.1.

SetDeskWallpaper

Описание:

BOOL FAR PASCAL SetDeskWallpaper(LPSTR lpszBmp);

Адрес: USER.285

Аргументы:

lpszBmp	Файл битовой карты «обоев» рабочей области окна.
---------	--

Возвращаемое значение:

TRUE в случае успеха.

Комментарии:

Функция загружает из файла и устанавливает в качестве текущих «обоев» рисунок, содержащийся в lpszBmp. Если значение параметра равно -1L, то загружаются «обои», по умолчанию определенные в файле Win.ini. В любом случае ни перерисовки фона, ни изменения файлов инициализации не производится.

В Windows 3.1 функция экспортируется под именем Bear285(), а все ее возможности реализуются документированной функцией SystemParametersInfo().

Версии: 3.0, 3.1.

SetEventHook

Описание:

```
void FAR PASCAL SetEventHook(FARPROC);
```

Адрес: USER.321

Комментарии:

Функция документирована только в Windows 3.1, но она существует и может быть вызвана также в версии 3.0.

Версии: 3.0, 3.1.

SetGridGranularity

Описание:

```
void FAR PASCAL SetGridGranularity(int nCell);
```

Адрес: USER.284

Аргументы:

nCell	Разрешение (в ячейках по восемь пикселей) решетки выравнивания расположения окон на рабочей поверхности или 0 для отмены выравнивания.
-------	--

Комментарии:

Для ускорения отображения окон Windows может выравнивать их границы по решетке с разрешением, кратным восьми пикселям.

В версии Windows 3.1 возможности этой функции реализованы одной из операций SystemParametersInfo().

Версия: 3.0.

SetInternalWindowPos

Описание:

```
void FAR PASCAL SetInternalWindowPos(HWND hWnd,  
    WORD wState, LPRECT lpRectWnd,  
    LPPOINT lpPointIcon);
```

Адрес: USER.461

Аргументы:

hWnd	Рабочее окно.
wState	Новое состояние окна: одна из констант SW_... .
lpRectWnd	Новые координаты окна на рабочей поверхности. Если NULL — сохраняются текущие.
lpPointIcon	Новые координаты для минимизированного окна. Если NULL — сохраняются текущие.

Комментарии:

Функция предоставляет возможности, аналогичные средствам функции SetWindowPlacement() в версии 3.1, но в то же время несколько более удобна и поддерживается также Windows 3.0.

Версии: 3.0, 3.1.

SetObjectOwner

Описание:

```
HANDLE FAR PASCAL SetObjectOwner(HANDLE hObject,  
    HANDLE hTask);
```

Адрес: GDI.461

Аргументы:

hObject	Дескриптор одного из объектов GDI.
---------	------------------------------------

hTask Задача — новый владелец hObject.

Возвращаемое значение:

Предыдущая задача-владелец hObject или NULL в случае неудачи.

Комментарии:

Функция инициализирует поле hOwner заголовка hObject в отладочной версии Windows 3.1, чем устанавливает владельца для соответствующего объекта и позволяет освобождать ресурсы при завершении задачи.

Версия: 3.1b.

SetPriority

Описание:

```
void FAR PASCAL SetPriority(HANDLE hTask,  
                           WORD wPrior);
```

Адрес: KERNEL.32

Аргументы:

hTask Изменяемая задача (0 — текущая).
wPrior Новый приоритет для задачи.

Комментарии:

Функция изменяет поле wPriority соответствующей структуры TASK, что изменяет приоритет исполнения hTask и положение соответствующей структуры TDB в системном списке.

Версии: 3.0, 3.1.

SetRelAbs

Описание:

```
int FAR PASCAL SetRelAbs(HDC hDC, int nRelAbs);
```

Адрес: GDI.5

Аргументы:

hDC	Рабочий контекст устройства.
nRelAbs	Текущий режим преобразования координат: 1 — если используются абсолютные координаты, 2 — относительные.

Возвращаемое значение:

Предыдущее значение режима преобразования или 0, если hDC недействителен.

Комментарии:

Функция устанавливает текущее поведение функций LineTo() и Polyline(): абсолютные координаты отсчитываются от начала координат контекста, относительные — от текущей позиции пера.

Версии: 3.0, 3.1.

SetSelectorBase

Описание:

```
WORD FAR PASCAL SetSelectorBase(WORD, DWORD);
```

Адрес: KERNEL.187

Комментарии:

Функция документирована только в Windows 3.1, но она существует и может быть вызвана также в версии 3.0.

Версии: 3.0, 3.1.

SetSelectorLimit

Описание:

```
WORD FAR PASCAL SetSelectorLimit(WORD, DWORD);
```

Адрес: KERNEL.189

Комментарии:

Функция документирована только в Windows 3.1, но она существует и может быть вызвана также в версии 3.0.

Версии: 3.0, 3.1.

SetSigHandler

Описание:

```
void FAR PASCAL SetSigHandler(FARPROC  
    lpfnHandler, FARPROC FAR *lpfpfnOldHandler,  
    BOOL FAR *lpbOldInst, WORD wInst, WORD  
    reserv);
```

Адрес: KERNEL.140

Аргументы:

lpfnHandler	Новый обработчик Ctrl-Break.
lpfpfnOldHandler	Адрес буфера для старого адреса обработчика.
lpbOldInst	В буфер помещается TRUE, если старый обработчик был обработчиком по умолчанию.

wInst	2, если устанавливается новый обработчик, 0, если восстанавливается обработчик по умолчанию.
reserv	Зарезервирован, должен быть равен 1.

Комментарии:

Функция позволяет текущей задаче устанавливать собственный обработчик для реакции на нажатие Ctrl-Break или имитировать данное событие функцией DoSignal() и одновременно узнавать адрес старого обработчика.

Версии: 3.0, 3.1.

SetSystemMenu

Описание:

```
BOOL FAR PASCAL SetSystemMenu (HWND hWnd, HMENU
                                hMenu) ;
```

Адрес: USER.280

Аргументы:

hWnd	Окно-клиент MDI.
hMenu	Новое системное меню для окна.

Возвращаемое значение:

TRUE в случае успеха.

Комментарии:

Функция позволяет менять системное меню *только* для окон-клиентов MDI.

Версии: 3.0, 3.1.

SetSystemTimer

Описание:

```
WORD FAR PASCAL SetSystemTimer(HWND hWnd,  
    int nIDEvent, WORD wElapse, FARPROC  
    lpTimerProc);
```

Адрес: USER.11

Комментарии:

Функция аналогична соответствующей документированной функции SetTimer(), но позволяет устанавливать таймеры уже после того, как ресурсы, доступные функции SetTimer(), кончаются. Вместо WM_TIMER функция использует для оповещения окон недокументированное сообщение WM_SYSTIMER.

Функция позволяет использовать таймеры, зарезервированные системой для работы в режиме максимальной загрузки, поэтому она должна вызываться *только* в случае крайней необходимости.

В версии 3.1 функция экспортируется под именем Bear11(), поэтому рекомендуется подключать ее не указывая имени.

Версии: 3.0, 3.1.

SetTaskQueue

Описание:

```
HANDLE SetTaskQueue(HANDLE hTask, HANDLE hQueue);
```

Адрес: KERNEL.34

Аргументы:

hTask	Изменяемая задача.
hQueue	Дескриптор структуры TASK_QUEUE.

Возвращаемое значение:

Дескриптор структуры TASK_QUEUE, ассоциированной с hTask до вызова функции.

Комментарии:

Функция позволяет ассоциировать с hTask структуру TASK_QUEUE, на которую указывает hQueue.

Версии: 3.0, 3.1.

ShrinkGDIHeap

Описание:

```
void FAR ShrinkGDIHeap();
```

Адрес: GDI.354

Комментарии:

Функция сокращает размер локального хипа GDI до минимально возможного размера, оставляя в то же время свободным не меньше килобайта.

Версии: 3.0, 3.1.

SnapWindow

Описание:

```
BOOL FAR PASCAL SnapWindow(HWND hWnd);
```

Адрес: USER.281

Аргументы:

hWnd Копируемое окно.

Возвращаемое значение:

TRUE в случае успеха.

Комментарии:

Функция сохраняет в Clipboard битовую карту, содержащую текущий образ части экрана, ограниченной рамкой окна hWnd.

Версия: 3.0.

SwitchToThisWindow

Описание:

```
void FAR PASCAL SwitchToThisWindow(HWND hWnd,  
    BOOL bRestore);
```

Адрес: USER.172

Аргументы:

hWnd	Окно, на которое необходимо переключиться.
bRestore	TRUE, если размеры минимизированного hWnd необходимо восстановить.

Комментарии:

Функция активизирует hWnd, передает ему фокус ввода, перерисовывает и, если необходимо, переводит из минимизированного в нормальное состояние.

Версии: 3.0, 3.1.

SysErrorBox

Описание:

```
int FAR PASCAL SysErrorBox(LPSTR lpszMsg,  
    LPSTR lpszTitle, WORD wButton1, WORD  
    wButton2, WORD wButton3);
```

Адрес: USER.320

Аргументы:

lpszMessage	Выдаваемое сообщение об ошибке.
lpszTitle	Заголовок сообщения.
wButton1	Тип левой кнопки (1).
wButton2	Тип средней кнопки (2).
wButton3	Тип правой кнопки (3).

Возвращаемое значение:

Номер кнопки, нажатой пользователем.

Комментарии:

Функция выводит на экран специальное окно сообщения о системной ошибке, содержащее от одной до трех кнопок с predetermined надписями. Возможны следующие типы кнопок: 1 — «OK», 2 — «Cancel», 3 — «Yes», 4 — «No», 5 — «Retry», 6 — «Abort», 7 — «Ignore». Если установлен старший бит значения типа, то кнопка становится кнопкой по умолчанию.

Функция реализована на очень низком уровне: она сохраняет работоспособность практически при любом состоянии системы, но в то же время при передаче неверных параметров может полностью вывести систему из строя.

Версии: 3.0, 3.1.

TileChildWindows

Описание:

```
void FAR PASCAL TileChildWindows(HWND hWnd
#ifdef WIN31
    ,WORD wAct
#endif
);
```

Адрес: USER.199

Аргументы:

hWnd	Окно, подокна которого должны быть переупорядочены.
wAct	Значение MDITITLE_SKIPDISABLED заставляет функцию не перемещать неактивные окна, MDITITLE_HORIZONTAL и MDITITLE_VERTICAL задают приоритет соответственно горизонтального или вертикального взаимного расположения окон, любое другое — игнорируется.

Комментарии:

Позволяет располагать в виде мозаики дочерние окна в не-MDI-приложениях, в том числе и для всей рабочей области окна вызовом TileChildWindows(GetDesktopWindow(), 0).

Обратите внимание, что число аргументов функции зависит от версии Windows. Неправильное задание аргументов приводит к общей ошибке защиты.

Версии: 3.0, 3.1.

UnicodeToAnsi

Описание:

```
int UnicodeToAnsi(LPSTR lpszUnicode, LPSTR  
    lpszAnsi);
```

Адрес: GDI.467

Аргументы:

lpszUnicode	Строка двухбайтных символов в стандарте Unicode. Длина строки не должна превышать 64К.
lpszAnsi	Буфер для преобразованной строки.

Возвращаемое значение:

Длина преобразованной строки в символах.

Комментарии:

Функция позволяет преобразовывать строки из формата Unicode в стандартный ANSI.

Версия: 3.1.

UserSeeUserDo

Описание:

```
DWORD FAR PASCAL UserSeeUserDo(WORD wAction,  
    WORD wParam1, WORD reserved, WORD wParam2);
```

Адрес: GDI.452

Аргументы:

wAction	Номер запроса к USER.
wParam1, wParam2	Данные, передаваемые с запросом.

Возвращаемое значение:

Если значение `wAction` не поддерживается функцией, то возвращается -1, иначе значение зависит от конкретного запроса.

Комментарии:

Функция дает возможность работы с локальным хипом USER. Допустимы следующие запросы:

1 (USUD_LOCALALLOC) — USER выделяет память в своем локальном хипе вызовом *LocalAlloc(wParam1, wParam2)* и возвращает дескриптор блока в младшем слове;

2 (USUD_LOCALFREE) — USER освобождает блок памяти в своем локальном хипе вызовом *LocalFree(wParam1)* и возвращает 0 в случае успеха;

3 (USUD_LOCALCOMPACT) — функция создает непрерывный свободный блок в локальном хипе USER, размеры которого передаются в `wParam2`, и возвращает максимально возможный размер такого блока;

4 (USUD_LOCALHEAP) — USER возвращает в младшем слове дескриптор своего локального хипа, который может быть использован для доступа к внутренним структурам USER и выделенным запросом `USUD_LOCALALLOC` этой функции блокам памяти;

5 (USUD_FIRSTCLASS) — USER возвращает в младшем слове дескриптор первой структуры CLASS в системном стеке.

Версия: 3.1.

WinExec

Описание:

WORD FAR PASCAL WinExec(LPSTR,int);

Адрес: KERNEL.166

Комментарии:

Функция документирована, но в документации не сказано, что возвращаемое значение, если оно больше 32, является дескриптором экземпляра (HINSTANCE) запущенной задачи.

Версии: 3.0, 3.1.

WriteOutProfiles

Описание:

```
void FAR PASCAL WriteOutProfiles();
```

Адрес: KERNEL.315

Комментарии:

Эта функция сбрасывает на диск кэш-буфер изменений в файлах инициализации.

Версии: 3.0, 3.1.

XCStoDS

Описание:

```
WORD FAR PASCAL XCStoDS();
```

Адрес: USER.315

Возвращаемое значение:

Селектор сегмента данных USER по умолчанию.

Комментарии:

USER является владельцем рабочей области экрана, и поэтому результат, аналогичный возвращаемому этой функцией, можно получить, вызвав функцию *GetWindowWord(GetDesktopWindow(), GWW_HINSTANCE)*.

Версия: 3.0.

Список литературы

1. **Schulman A., Maxey D., Pietrek M.** Les Coulisses de Windows (Undocumented Windows). — Paris: Editions Addison-Wesley France, 1992. — 702 p.
2. **Bar J., Bauder I.** Le Grand Livre de la Programmation Windows 3.1. — Paris: Data Becker GmbH, Düsseldorf; Micro Application, 1992. — 908 p.
3. **Borland C++ 3.0 Programmer's guide.** — US, CA: Borland International, 1991. — 467 p.
4. **Borland C++ 3.1 Windows API online documentation.** — US, CA: Borland International, 1992.
5. **Справочник Windows 3.0 для программистов.** — М.: ППИ НЦ, 1991. — 458 с.
6. **Гладков С.А., Фролов Г.В.** Программирование в Microsoft Windows. — М.: Диалог-МИФИ, 1992. — 604 с.
7. **Turbo Assembler 3.0. Quick reference Guide.** — US, CA: Borland International, 1991. — 148 p.
8. **Shulman A., Michels R.J. et al.** Undocumented DOS. — Paris: Addison-Wesley Publishing Company, Inc., 1991. — 682 p.
9. **Борковский А.Б., Зайчик Б.И., Боровикова Л.И.** Словарь по программированию. — М.: «Русский язык», 1991. — 286 с.
10. **Шулман Э.** Недокументированные функции Windows. // PC Magazine Russian Edition. — 1992. — No. 3. — P. 114—123; No. 4 — P. 97—105.
11. **Дункан Р.** Библиотека ToolHelp для Windows 3.1. // PC Magazine Russian Edition. — 1992 — No. 5. — P. 98—103; No. 7 — P. 113-120; No. 8. — P. 106-111.
12. **Зубанов Ф.** Microsoft Windows: открываем тайны. // Компьютер пресс. — 1992. — № 6. — С. 4—14; № 7. — С. 55—61.

Приложение

Список функций, структур, переменных и сообщений по типам

Управление селекторами

__0000H	IsSharedSelector
__0040H	PrestoChangoSelector
__ROMBIOS	SelectorAccessRights
AllocAlias	SELTAB
AllocCStoDSAlias	SetSelectorBase
AllocSelectorArray	SetSelectorLimit

Управление окнами

CalcChildScroll	WM_BEGINDRAG
CascadeChildWindows	WM_ENTERSIZEMOVE
CLASS	WM_EXITSIZEMOVE
FillWindow	WM_ISACTIVEICON
GetInternalWindowPos	WM_LBTRACKPOINT
GetNextQueueWindow	WM_PAINTICON
ScrollChildren	WM_SETVISIBLE
SetInternalWindowPos	WM_ACTIVATESHELLWINDOW
SwitchToThisWindow	WM_OTHERWINDOWCREATED
TileChildWindows	WM_OTHERWINDOWDESTROYED
WND	

Общая информация и управление системой

ATOMTABLE	GetSelectorBase
ATOMENTRY	GetSelectorLimit
ControlPanelInfo	GlobalMasterHandle
DeletePathName	IsROMFile
ExitKernel	IsROMModule
FileCdr	IsUserIdle
FlushCachedFileHandle	IsValidMetaFile
Get80x87SaveSize	RCos
GetAppCompatFlags	RSin
GetCodeHandle	THHOOK
GetExePtr	THHook
Save80x87SaveState	
SetDeskPattern	
SetDeskWallPaper	
SetGridGranularity	
UnicodeToAnsi	

Системные вызовы

BroadcastMessage	DOS3Call
DisableDOS	DoSignal
DisableKernel	fTrapping0
EnableDOS	GetFilePortName
EnableKernel	GetLpErrMode

GetMouseEventProc	SetSigHandler
GetSetKernelDOSProc	SysErrorBox
Keybd_Event	THUNKS
NoHookDOSCall	WriteOutProfiles
Restore80x87SaveState	WM_ALTTABACTIVE
SetEventHook	

Работа с таймерами

CreateSystemTimer	KillSystemTimer
DisableSystemTimers	SetSystemTimer
EnableSystemTimers	WM_SYSTIMER
GetTimerResolution	

Общее управление графическим устройством

Death	IconSize
DisableOEMLayer	RepaintScreen
EnableOEMLayer	Resurrection

Отладка

DiagOutput	RegisterPTrace
DiagQuery	

Управление задачами

DirectedYield	GdiSeeGdiDo
---------------	-------------

GetCurrentPDB	IsWinOldAppTask
GetCurrentTask	LockCurrentTask
GetExeVersion	LockMyTask
GetExpWinVer	MODULE
GetModuleHandle	SetTaskQueue
GetQueueStatus	TASK_QUEUE
GetTaskDS	TDB
GetTaskQueue	UserSeeUserDo
GetTaskQueueDS	WinExec
GetTaskQueueES	WM_GETHOTKEY
INSTANCE	WM_SETHOTKEY
IsTaskLocked	XCSToDS

Работа с ресурсами

BITMAPOBJ	IsGDIObject
BRUSHOBJ	LoadCursorIconHandler
CURSOR	LoadDIBCursorHandler
CURSORICONINFO	LoadDIBIconHandler
DirectResAlloc	LoadIconHandler
DumpIcon	MakeObjectPrivate
FONTOBJ	PALETTEOBJ
GetIconId	PENOBJ
GDIOBJHDR	SetObjectOwner
ICON	

Протокол Drag&Drop

DragDetect	WM_DRAGMOVE
DRAGINFO	WM_DRAGSELECT
DragObject	WM_DROPOBJECT
DROPINFO	WM_QUERYDROPOBJECT
WM_DRAGLOOP	

Работа с меню

EndMenu	SetSystemMenu
LookupMenuHandle	WM_EXITMENULOOP
MENU	WM_ENTERMENULOOP
MENUITEM	

Работа с контекстом устройства и графика

DC	IntersectVisRect
DCE	IsDCCurrentPalette
DCSAVE	IsDCDirtyOffsetVisRgn
ExcludeVisRect	PaintRect
FastWindowFrame	RealizeDefaultPalette
GetClipRgn	RestoreVisRgn
GetControlBrush	SaveVisRgn
GetCurLogFont	SelectVisRgn
GetDCState	SetDCOrg
GetPhysicalFontHandle	SetDCState
GetRelAbs	SetDCStatus

SetRelAbs	SnapWindow
-----------	------------

Работа с памятью

__WINFLAGS	IsBadStringPtr
BURGERMASTER	IsBadWritePtr
FarGetOwner	LOCAL_ARENA
FarSetOwner	LOCAL_HEAP
GetFreeMemInfo	LocalCountFree
GetHeapSpaces	LocalHandlerDelta
GLOBAL_ARENA	LocalHeapSize
GlobalFreeAll	LocalNotify
GlobalHandleNoRIP	LongPtrAdd
InitAtomTable	SchrinkGDIHeap
IsBadReadPtr	

Содержание

Предисловие.	3
Часть I. Внутренние структуры Windows	9
ATOMTABLE.....	9
ATOMENTRY.....	10
BITMAPOBJ.....	10
BRUSHOBJ	11
BURGERMASTER.....	11
CLASS	14
CURSOR.....	15
CURSORICONINFO.....	15
DC	16
DCE.....	21
DCSAVE.....	22
DRAGINFO	22
DROPINFO.....	23
FONTOBJ.....	24
GDIOBJHDR.....	25
GLOBAL_ARENA	26
ICON.....	28
INSTANCE.....	28
LOCAL_ARENA	30
LOCAL_HEAP	31
MENU.....	34
MENUITEM	35
MODULE.....	36

PALETTEOBJ.....	42
PENOBJ.....	43
SELTAB.....	43
TASK_QUEUE.....	44
TDB.....	47
THUNKS.....	50
THHOOK.....	51
WND.....	53
Часть II. Сообщения Windows	56
WM_ACTIVATESHELLWINDOW	56
WM_ALTTABACTIVE.....	56
WM_BEGINDRAG	57
WM_DRAGLOOP.....	57
WM_DRAGMOVE.....	58
WM_DRAGSELECT.....	58
WM_DROPOBJECT	59
WM_ENTERMENULOOP	59
WM_ENTERSIZEMOVE.....	60
WM_EXITMENULOOP	60
WM_EXITSIZEMOVE.....	60
WM_GETHOTKEY.....	61
WM_ISACTIVEICON.....	61
WM_LBTRACKPOINT.....	62
WM_OTHERWINDOWCREATED	62
WM_OTHERWINDOWDESTROYED	63
WM_PAINTICON.....	63

WM_QUERYDRPOBJECT	63
WM_SETHOTKEY	64
WM_SETVISIBLE	65
WM_SYSTIMER	65
Часть III. Экспортируемые переменные	66
__0000H	66
__0040H	66
__ROMBIOS	67
__WINFLAGS	67
fTrapping0	68
THHook	68
Часть IV. Функции Windows	69
AllocAlias	69
AllocCStoDSAlias	70
AllocSelectorArray	70
BroadcastMessage	71
CalcChildScroll	71
CascadeChildWindows	72
ControlPanelInfo	73
CreateSystemTimer	74
Death	75
DeletePathName	76
DiagOutput	76
DiagQuery	77
DirectedYield	77
DirectResAlloc	78

DisableDOS.....	78
DisableKernel	79
DisableOEMLayer	79
DisableSystemTimers.....	80
DOS3Call	80
DoSignal.....	80
DragDetect	81
DragObject	82
Dumplcon	83
EnableDOS.....	84
EnableKernel	84
EnableOEMLayer	85
EnableSystemTimers.....	85
EndMenu	86
ExcludeVisRect.....	86
ExitKernel	87
FarGetOwner.....	87
FarSetOwner.....	88
FastWindowFrame	89
FileCdr	89
FillWindow	91
FlushCachedFileHandle	92
GdiSeeGdiDo.....	92
Get80x87SaveSize	93
GetAppCompatFlags	94

GetClipRgn	94
GetCodeHandle	95
GetControlBrush	96
GetCurLogFont	96
GetCurrentPDB	97
GetCurrentTask	97
GetDCState	97
GetExePtr	98
GetExeVersion	99
GetExpWinVer	99
GetFilePortName	100
GetFreeMemInfo	100
GetHeapSpaces	101
GetIconId	102
GetInternalWindowPos	102
GetLpErrMode	103
GetModuleHandle	103
GetMouseEventProc	104
GetNextQueueWindow	105
GetPhysicalFontHandle	105
GetQueueStatus	106
GetRelAbs	106
GetSelectorBase	107
GetSelectorLimit	107
GetSetKernelDOSProc	107
GetTaskDS	108

GetTaskQueue	109
GetTaskQueueDS	109
GetTaskQueueES	109
GetTimerResolution	110
GlobalFreeAll	110
GlobalHandleNoRIP	111
GlobalMasterHandle.....	111
IconSize	112
InitAtomTable	112
InquireVisRgn.....	113
IntersectVisRect.....	113
IsBadReadPtr.....	114
IsBadStringPtr	114
IsBadWritePtr	115
IsDCCurrentPalette	115
IsDCDirty	116
IsGDIObject.....	116
IsROMFile	117
IsROMModule	117
IsSharedSelector	118
IsTaskLocked.....	118
IsUserIdle	119
IsValidMetaFile	119
IsWinOldAppTask.....	120
Keybd_Event	120
KillSystemTimer (I).....	121
KillSystemTimer (II)	121

LoadCursorIconHandler	122
LoadDIBCursorHandler	123
LoadDIBIconHandler.....	123
LoadIconHandler.....	124
LocalCountFree.....	124
LocalHandlerDelta	125
LocalHeapSize.....	125
LocalNotify	126
LockCurrentTask	126
LockMyTask.....	127
LongPtrAdd	127
LookupMenuHandle.....	128
MakeObjectPrivate	128
NoHookDOSCall	129
OffsetVigRgn	130
PaintRect.....	130
PrestoChangoSelector	131
RCos	132
RealizeDefaultPalette.....	132
RegisterPTrace.....	133
RepaintScreen	134
Restore80x87State	135
RestoreVisRgn	135
Resurrection	136
RSin	136
Save80x87State	137
SaveVisRgn.....	137

ScrollChildren	138
SelectorAccessRights	138
SelectVisRgn.....	140
SetDCOrg.....	140
SetDCState.....	141
SetDCStatus	141
SetDeskPattern	142
SetDeskWallpaper.....	143
SetEventHook	144
SetGridGranularity	144
SetInternalWindowPos.....	145
SetObjectOwner.....	145
SetPriority	146
SetRelAbs	147
SetSelectorBase	147
SetSelectorLimit	148
SetSigHandler.....	148
SetSystemMenu.....	149
SetSystemTimer.....	150
SetTaskQueue	150
ShrinkGDIHeap	151
SnapWindow.....	151
SwitchToThisWindow	152
SysErrorBox.....	153
TileChildWindows	154
UnicodeToAnsi.....	155
UserSeeUserDo	155

WinExec	156
WriteOutProfiles	157
XCStoDS	157
Список литературы.	159
Приложение. Список функций, структур, переменных и сообщений по типам	160

Акционерное общество

"РиС"

маркетинговая фирма

ИЗДАТЕЛЬСТВА "РАДИО И СВЯЗЬ"

Р е а л и з у е т

ОПТОМ И В РОЗНИЦУ

ПО ИНДИВИДУАЛЬНЫМ ЗАКАЗАМ

радиолобительскую литературу, а также учебную, справочную, научную, производственно-техническую и научно-популярную литературу по информатике, вычислительной технике, радиоэлектронике и связи.

Доставку заказа может выполнить отделение "Книга-почтой" АО "РиС".

Отделение "Книга-почтой" выполняет

- ▶ оптовый заказ после предварительной оплаты
- ▶ индивидуальный — наложенным платежом

Заказы просим направлять по адресу:

103473, Москва, 2-й Щемилковский, 4/5,

Акционерное общество "РиС"

телефон для справок (095) 258-72-57

(095) 313-83-45

факс (095) 258-72-57

**В издательстве "Радио и связь"
в 1994 г. выйдут в свет книги:**

Мячев А.А., Алексеев Е.С.
Интерфейсы и сети ЭВМ.
Англо-русский толковый словарь

Под ред. Кнеллера М.И.
AutoCAD. Полезные рецепты

Лямин Л.В.
Макроассемблер MASM

Иванов В.П., Батраков А.С.
Трехмерная компьютерная графика

Шкаев А.В.
Настольные издательские системы

Судовцев В.А.
Терминология по электросвязи
Англо-русский словарь

Уилтон Р.
Программирование видеосистем
персональных компьютеров

Хаусли Т.
Системы передачи данных

"РАДИО И СВЯЗЬ"